uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería en Tecnologías de Telecomunicación
Curso Académico: 2017/2018

MONITORIZACIÓN DE LA ACTIVIDAD DE SUJETOS EN UNA CASA INTELIGENTE EMPLEANDO TÉCNICAS DE DATA ANALYTICS

Autor:
Ana Galán Hernández

Tutor:
Julio Villena Román
Madrid, 2 de Julio de 2018

"Vive como si fueras a morir mañana.
Aprende como si fueras a vivir siempre"

Mahatma Gandhi

Agradecimientos

Al terminar este Trabajo Fin de Grado he llegado a la conclusión de que no solo es el final del grado universitario, en el que tanto esfuerzo y dedicación he puesto, sino también el final de un ciclo, que he compartido con gente maravillosa y a la que tengo que agradecer que me hayan ayudado a graduarme.

En primer lugar, me gustaría dar las gracias a mis padres y a mi hermana por darme la oportunidad de formarme, por dejar que buscara mi propio camino, pero sobre todo por apoyarme y darme todo el cariño que he necesitado. También me gustaría dar las gracias a mis abuelos, Aurora y Juan José, por acordarse y llamarme después de cada uno de mis exámenes.

En segundo lugar, quiero agradecer a mis compañeros y amigos de la carrera que me hicieran las clases, comidas, trabajos y prácticas más divertidas y amenas. Y sobre todo a mi novio Fernando que me ha sabido comprender y darme el ánimo, la tranquilidad y el aliento que necesitaba en los peores momentos.

Por último, pero no menos importante, quiero dar las gracias a la Universidad, a todos los profesores y trabajadores que han proporcionado un ambiente de aprendizaje excepcional. Y en exclusiva a mi tutor Julio Villena Román por darme la oportunidad de hacer este trabajo, su paciencia, disponibilidad, generosidad e inspirarme como profesional.

Resumen

El objetivo principal de este Trabajo Fin de Grado es evaluar distintos modelos que predicen la actividad diaria de una persona en una casa inteligente, a partir de datos recopilados de un conjunto de sensores. Este modelo estaría pensado como base para un sistema de monitorización que pueda servir de apoyo a personas mayores o con movilidad reducida, de modo que se les facilite llevar una vida más independiente y segura.

Para dicha evaluación se han analizado dos escenarios con dos conjunto de datos distintos. El primero de ellos, lo proporciona la competición *Senior Data Science: Safe Aging with SPHERE* [1] a través de la plataforma de DrivenData [2], y el segundo el repositorio UCI [3]. En ambos conjuntos de datos se busca reconocer las actividades que están haciendo los sujetos analizados, pero en el primero se utiliza la información que se recibe de los sensores de bajo coste instalados en casas inteligentes y en el segundo se utiliza la información que se recibe de los sensores integrados en un Smartphone. Las actividades que se reconocen con cada conjunto de datos son las siguientes:

Actividades del primer conjunto de datos:

- Subir escaleras
- Bajar escaleras
- Saltar
- Transportar algo
- Caminar
- Inclinado
- De rodillas
- Tumbado
- Sentado
- En cuclillas
- De pie
- De pie inclinado
- De rodillas a de pie
- Tumbado a sentado
- Sentado a tumbado
- Sentado a de pie
- De pie a de rodillas
- De pie a de rodillas
- De pie a sentado
- Girarse

Actividades del segundo conjunto de datos:

- Caminar
- Subir escaleras
- Bajar escaleras
- Sentado
- De pie
- Tumbado
- De pie a sentado
- De sentado a de pie
- De sentado a tumbado
- De tumbado a sentado
- De pie a tumbado
- De tumbado a de pie

El análisis predictivo de ambos conjuntos de datos tiene el mismo fin, pero el primero, utiliza la información que se obtiene de diferentes sensores distribuidos por las estancias de una casa inteligente; y el del segundo contribuye como complemento, para valorar el rendimiento de los sensores del Smartphone.

En el capítulo 4 se detallan el pre-procesado y los diferentes experimentos que se han llevado a cabo para el análisis predictivo, sobre el primer conjunto de datos. Algunas de las técnicas de pre-procesado de los datos que se han aplicado para obtener buenos resultados al hacer el análisis predictivo son:

- Integración de datos
- Limpieza de datos
- Generación y eliminación de atributos
- Transformación de datos

Tras el pre-procesado de datos, se realizan diferentes experimentos distribuidos en tres fases. En una primera fase se utiliza la herramienta RapidMiner con diferentes familias de algoritmos, para evaluar cuál se adapta mejor. Tras esto, se concluye que el algoritmo con el que se obtiene el resultado óptimo es *Gradient Boosted Trees*, con un *accuracy* de 74,78 %.

En una segunda fase, se ha utilizado una implementación del mismo algoritmo en Python y se ha obtenido un *accuracy* similar. La competición a la que pertenecen estos datos está cerrada, por tanto, no es posible publicar las predicciones para evaluarlas y comparar los resultados con otros participantes. Para hacerse una idea de la calidad de nuestro clasificador con respecto a otros participantes, se simula una función para calcular el *Brier Score* según los criterios establecidos. Al aplicar dicha función se obtiene un resultado de 16,6 %; este resultado se puede considerar que es suficiente bueno, ya que, al compararlo con el ranking público de la competición [4], se ha alcanzado la posición 20 de 579.

En una tercera fase, se ha optado por reducir la dimensionalidad de los datos para intentar mejorar los resultados, pero a medida que se reduce la dimensionalidad, los resultados empeoraron.

Por otro lado, en el capítulo 5 se detallan las variables y los experimentos realizados sobre el segundo conjunto de datos utilizando la herramienta RapidMiner, se ha obtenido el mejor resultado con el modelo *W-SimpleLogistics*, alcanzando un *accuracy* de 94,02 %. Este resultado obtenido es lo suficientemente bueno, como para que se incluya la tecnología de los Smartphone en cualquier diseño de sistema de monitorización, como el que se plantea en el capítulo 6.

La función del sistema de monitorización que se propone, sería reconocer el movimiento de los individuos y determinar si algunas de las actividades realizadas por el sujeto son peligrosas. Una actividad que podría accionar el sistema de alerta y necesitar intervención presencial es, por ejemplo, que el sujeto este tumbado en el baño un tiempo prolongado.

Extended Abstract

The recent diffusion of cognitive computing has fostered the emergence of new technologies, which provide machines with artificial intelligence the capacity to perceive the world and help people in different areas of their daily lives [5]; giving them more autonomy and allowing the development of our society.

In addition, the rise of social networks, smartphones, digital cameras and other digital devices have simplified the data extraction, which has led to Big Data environments. In this type of environment, it is important to take into account the amount of information, the speed with which the data are generated, and also the variety and the veracity of the same, so that the results show the most exhaustive reality .

The interconnection of devices has led to solutions based on the Internet of Things (IoT) that allows to collect and promote timely information [6]. There are many IoT solutions that make life easier for individuals, such as controlling windows, doors, lights, gas.

The main objective of this Bachelor's Degree Thesis is to evaluate different models that predict the daily activities of a person in a smart home, from data collected from a set of sensors. This model would be thought as the basis for a monitoring system that can serve as support for the elderly or for those with reduced mobility, so that they can lead a more independent and secure life.

For this evaluation, two data sets have been analyzed, the first one is provided by the competition *Senior Data Science: Safe Aging with SPHERE* [1] through the DrivenData platform, and the second by the UCI [3] repository. Both data sets seek to recognize the activities, but in the first one, the information is received from low-cost sensors that are installed in smart homes, while in the second one, the information came from the sensors integrated in a Smartphone.

Data mining is defined as the process of extracting knowledge from the data. There are several and diverse methodologies to carry out data mining projects,. Two of the most common ones are: SEMMA (Sample, Explore, Modify, Model, Assess) and CRISP-DM (CRoss-Industry Standard *Process* for Data Mining). In this project CRISP-DM methodology is used, because its first stage is related to the setting of the more appropriate goals, which is a very important step in any investigation. There are three types of tasks in data mining: descriptive, predictive

and prescriptive. Descriptive tasks allow to understand past behavior and future results. Predictive tasks allow to extract the necessary information to provide good estimations; and prescriptive ones provides a series of tips to reach a concrete result.

Another important part of data mining and artificial intelligence is machine learning, which allows a machine to learn by itself with the information it receives from different devices and with the support of different algorithms and techniques. Most algorithms can be classified according to their type of learning as supervised learning, unsupervised learning or semi-supervised learning.

On the one hand, in supervised learning each instance has to have a variable labeled, some examples of these algorithms are: neural networks, naive bayes classifiers, kNN, decision trees, among many others. On the other hand, with non-supervised learning it is not necessary to have labels, some examples of this type are: clustering algorithm and principal component analysis (PCA). Finally, semi-supervised learning uses both, tagged data and untagged data. It can be seen as a combination of the other two types.

First data set

In the first data set a total of 20 activities are distinguished and each of them is defined with a tag, as it is specified in the table 1.

Label	**Activity**	**Label**	**Activity**
a_ascend	ascend stairs	*p_stand*	standing
a_descend	descend stairs	*t_bend*	stand-to-bend
a_jump	jump	*t_kneel_stand*	kneel-to-stan
a_loadwalk	walk with load	*t_lie_sit*	lie-to-sit
a_walk	walk	*t_sit_lie*	sit-to-lie
p_bent	bending	*t_sit_stand*	sit-to-stand
p_kneel	kneeling	*t_stand_kneel*	stand-to-knee
p_lie	lying	*t_stand_sit*	stand-to-sit
p_sit	sitting	*t_straighten*	bend-to-stand
p_squat	squatting	*t_turn*	turn

Table 1: Description of the 20 activities

The data of this set is distributed in six files with CSV format: *targets.csv*, *pir.csv*, *acceleration.csv*, *video_hallway.csv*, *video_living_room.csv*, and *video_kitchen.csv*, each of them has a different structured time period. It has been necessary to integrate and unify them during the pre-processing phase of the data.

To integrate and unify them, the attribute *t_round* has been created, which specifies the time stamp of the probability set of each activity. After that, different techniques have been applied, which have allowed to clean the data of unknown data (NaNs), eliminating the attributes that did not provide information, generating new attributes and transforming the data. One of the techniques that has been used is One Hot Encoding, to transform a nominal variable into as many binary variables as categories has. Some of the variable types that have been added are: temporary variables, mobile variables, varibles differentials, trend variables or probabilistic variables.

Experiments performed in the predictive analysis of the first data set can be divided into three phases. In a first phase the RapidMiner platform is used with different algorithm families to evaluate which algorithm yields the best result. Some of the algorithms that have been evaluated are: *K-Nearest Neighbours*, *Decision Tree*, *Random Forest*, *Gradient Boosted Trees*, *Rule Induction* and *Deep Learning*.

- *K-Nearest Neighbours* belongs to the family of Lazy classifiers Learners and is very sensitive to its two main parameters: k and *measure type*; the number of nearest neighbors and the type of metric that is used to calculate the proximity of the nearest k neighbors.
- *Decision Tree* is considered a non-parametric type of method because, a priori, it does not presuppose either the volume of the class or the tree structure. They are also very intuitive and minimize the risk and maximize the *accuracy*.
- *Random Forest* is a type of *Ensemble Tree* that combines different trees with random and independent starts. The number of trees is specified with the parameters *number of trees*. For each tree a subset of samples are selected.
- *Gradient Boosted Trees* is another type of *Tree Assemble* that combines different trees that get results through gradually improved estimations. It predicts by simple majority and does not take into account the results that have been obtained previously. Therefore, the trees are independent.
- *Rule Induction* is a simple algorithm based on rules.
- *Deep Learning* is a neural network based on a multi-layered artificial feed-forward neural network that is trained with a stochastic gradient descent by retro-propagation.

To optimize the parameters of each of the algorithms, the technique of cross-validation is used during the training phase. This phase concludes that the algorithm with which the optimal result are obtained is *Gradient Boosted Trees* with an *accuracy* of 74.78 %.

In a second phase, this algorithm has been implemented in Python and a similar *accuracy* has been obtained. The competition to which these data belong is closed, therefore it is not possible to publish the predictions to evaluate them and

compare the results with other participants. To get an idea of the quality of our classifier with respect to other participants, a function is simulated to calculate the *Brier Score* according to the established criterion. When applying this function, a result of 16.6 % is obtained; this result can be considered to be good enough, since, it would correspond to the position 20th of 579 in competition ranking [4].

In a third phase, we tried to reduce the dimensionality of the data in order to improve the results, but as the dimensionality is reduced, the results become worse.

Second data set

The second data set is distinguished by 12 activities and each of them is defined with a label, as specified in the table. Moreover, it was already preprocessed, so it has not been necessary to carry out a procedure to integrate and unify it.

Type of activity	**Label**	
	Nominal	**Numeric**
Dynamic	*WALKING*	1
	WALKING_UPSTAIRS	2
	WALKING_DOWNSTAIRS	3
Static	*SITTING*	3
	STANDING	5
	LAYING	6
Transitory	*STAND_TO_SIT*	7
	SIT_TO_STAND	8
	SIT_TO_LIE	9
	LIE_TO_SIT	10
	STAND_TO_SIT	11
	LIE_TO_STAND	12

Table 2: Description of the 12 activities

Dduring its predictive analysis some of the most remarkable algorithms have been: *K-Nearest Neighbours*, *Naive Bayes*, *Gradient Boosted Trees*, *Deep Learning*, *W-Simple Logistics* and *W-REPTree*.

- *Naive Bayes* is a type of classifiers based on *Bayes Theorem*, that predict probabilities by assuming conditional class independence.
- *W-Simple Logistics* is a classifier that constructs linear logistic regression models with the algorithm of *boosting*, *LogitBoost*.
- *W-REPTree* is a faster decision tree, in which the unknown values are handled by dividing the corresponding instances into parts.

The best results were obtained with the *W-SimpleLogistics* algorithm, reaching an *accuracy* of 94.02 %. Therefore, this result can be consider good enough for the Smartphone technology could be included in the design of the monitoring system that are posed next.

Monitoring system

The monitoring system proposed in this project provides greater security in houses for people with reduced mobility, so that they can lead a life with more autonomy and independence [7]. This system would be responsible for recognizing the position of the subjects and classify it as dangerous or not.

This system would be in charge of recognizing the position of the subjects, through a predictive analysis with the information received from the different electrical sensors previously installed in the homes of the subjects. And then it classifies them as **DANGER**. However, activities that are not considered dangerous are classified with the label **OK**.

In case of classifying any of these activities as **DANGER**, something that is strange, risky or anomalous is supposed to be happening, an alert would be activated and a monitoring center would be contacted. For example, if the system detects that the subject is lying in the bathroom for a long time, the alert will be activated, classifying the behavior as anomalous. This system would give autonomy and security to people with reduced mobility, respecting their privacy.

The technical design for the monitoring system consists of five modules and requires the installation of environmental sensors, access points and cameras around the house. In addition, the use of the smartphone has been included regarding the good results obtained in the analysis of the second data set.

- *Extraction of data.* During this phase the data coming from the sensor or devices distributed by the rooms of the smart home are collected.
- *Data preprocessing.* After the extraction of data it is necessary to preprocess them to achieve good results in the predictive analysis.
- *Predictive analysis.* In this phase the movements of the users according to the received data are predicted.
- *Danger detection.* During this phase, we evaluate whether the movements that have been predicted are dangerous or not.
- *Communicative intervention.* This phase is activated if any dangerous activity is detected.

To finalize with, the monitoring system could have other improvements as the posibility to control if individual take their medication and being able to detect smoke or gas inside the smart home.

Índice general

Índice de tablas

Índice de figuras

1 | Planteamiento del Trabajo Fin de Grado

1.1. Introducción

Gracias a la reciente difusión de la computación cognitiva, se ha impulsado el surgimiento de nuevas tecnologías, que permiten a las máquinas percibir el mundo; mediante la recopilación, el análisis y la comprensión de datos, con el objetivo de ayudar a las personas en distintos ámbitos, de su vida diaria [5]; proporcionando más autonomía y permitiendo el crecimiento de nuestra sociedad.

De esta manera, se pretende dotar a las máquinas de inteligencia usando técnicas de *data analytics*, que simulen procesos de pensamiento humano. Una de las características más importantes de los sistemas cognitivos es la capacidad de aprendizaje, que hace posible que las máquinas sean autodidactas y no sea necesario programar todas las reglas con antelación, con la ventaja constante de mejorar sus resultados.

Por otro lado, el auge de las redes sociales, los smartphones, las cámaras o los sensores ha permitido que el volumen, la velocidad y la facilidad de extracción de datos haya aumentado y mejorado; debido a que los datos son generados automáticamente por máquinas. Otras características importantes a destacar en este tema son, la variedad y la veracidad de los datos. Todo esto se encuentra recogido en las llamadas 3V's del *Big Data*, volumen, velocidad y variedad [8]. Estas V's se pueden extender con otras como la veracidad y el valor [9], haciendo en conjunto referencia a los principales retos a los que se intenta hacer frente con las tecnologías englobadas bajo el término *Big Data*.

Con el objetivo de superar estos retos se han desarrollado diferentes procesos de limpieza y técnicas de análisis y pre-procesado, que ha simplificado el sesgo y ha reducido el ruido y la alteración de datos, extendiendo la posibilidad de usar no solo datos estructurados si no también datos no estructurados.

Todo esto, está íntimamente relacionado con el concepto o tecnologías del llamado *Internet of Things* (IoT), al que muchos consideran como la nueva revolución industrial, capaz de dar un giro de 180 grados a la economía mundial [6]. Este concepto hace referencia a la idea de un mundo en el que todos los dispositivos electrónicos estén interconectados y compartan información de forma que se pueda

mejorar la calidad de vida de la población.

Este Trabajo Fin de Grado se desarrolla con el objetivo de evaluar el reconocimiento de las actividades de individuos mediante un análisis predictivo con la información que recibe de los diferentes sensores.

Para estudiar el rendimiento de los diferentes modelos de algoritmos predictivos se utilizan los datos de la competición *Senior Data Science: Safe Aging with SPHERE* [1] de la plataforma DrivenData [2]; y un segundo conjunto de datos del repositorio UCI [3], que recibe la información de la actividad de los sujetos, de los sensores integrados en un Smartphone.

1.2. Motivación del trabajo

La principal motivación de este Trabajo de Fin de Grado nace del creciente interés por el bienestar y la saludo que se está viviendo hoy en día, junto con la proliferación de dispositivos, sensores y técnicas de aprendizaje máquina e inteligencia artificial que ayudan a detectar patrones y extraer información de grandes cantidades de datos. En este contexto, se plantea la utilización de distintas técnicas de inteligencia artificial, sobre los datos extraídos a partir de un sistema de sensores desplegados por una vivienda.

1.3. Objetivos

El objetivo principal de este Trabajo Fin de Grado es evaluar distintos modelos que predicen la actividad diaria de una persona en una casa inteligente, a partir de datos recopilados de un conjunto de sensores.

Este modelo estaría pensado como base para un sistema de monitorización que pueda servir de apoyo a personas mayores o con movilidad reducida. Lo que implicaría que dichos individuos pudiesen llevar una vida más independiente y segura. Este sistema de monitorización analizaría cada una las actividades y detectaría las peligrosas, como por ejemplo, que el individuo esté mucho tiempo tumbado en el baño.

Otro de los objetivos secundarios que se han buscado con el desarrollo de este trabajo es la adquisición de nuevos conocimientos y técnicas de *data analytics*.

1.4. Estructura de la memoria

Esta memoria se encuentra estructurada en 8 capítulos, con los siguientes títulos:

- **Planteamiento del Trabajo Fin de Grado**: en este primer capítulo se realiza una breve introducción y se exponen las principales motivaciones y objetivos del TFG.
- **Estado del arte**: contextualizará de una forma tecnológica, social, económica y legal el fondo sobre el que se trabajará.
- **Ciencia de datos**: en este capítulo, se organiza y estructura la metodología y los conceptos necesarios para abordar el problema en cuestión.
- **Escenario 1**: se detalla un análisis descriptivo y predictivo del primer conjunto de datos con el que se trabaja.
- **Escenario 2**: se detalla un análisis descriptivo y predictivo del segundo conjunto de datos con el que se trabaja.
- **Conclusiones y líneas futuras**: se exponen los resultados y conclusiones del trabajo; junto con un planteamiento del sistema de monitorización y posibles mejoras futuras.
- **Presupuesto**: en este capítulo se detalla la planificación y la estimación del presupuesto requerido.
- **Bibliografía**: en este último capítulo se detallan las referencia utilizadas a lo largo del TFG.

2 | Estado del arte

2.1. ¿Qué es "Internet de las cosas"?

El concepto de "Internet de las cosas" (IoT) se refiere a una red con componentes electrónicos, sensores, conectividad de red y software, integrado con el mundo físico [10]. Las soluciones de IoT permiten recopilar todo tipo de información desde algunos bytes hasta varios megabytes, según los requisitos de la aplicación. Porque a diferencia de las aplicaciones de la telemetría[1] tradicional, estas soluciones pueden proporcionar datos a los usuarios solo cuando son requeridos por la situación [11] [12].

Uno de los principales objetivos de los servicios de atención domiciliaria es ayudar a las personas mayores o que tienen algún tipo de discapacidad a ser independientes proporcionándoles diversas formas de apoyo para facilitarles la vida en sus propios hogares el mayor tiempo posible. Por tanto, IoT es una forma revolucionaria que brinda oportunidades de vida independiente y de ayudar a mejorar la calidad de vida de los individuos.

Algunas de las soluciones IoT son la posibilidad de controlar las puertas delanteras, las luces, el gas... Esto proporciona una mejor calidad de vida y tranquilidad para los cuidadores. Teniendo en cuenta que la población es cada vez más anciana y el número de discapacitados aumenta, es importante tener en cuenta que con el uso de la tecnología móvil y la integración inalámbrica se puede reducir significativamente el coste de adaptar los hogares [11].

2.2. Entorno socio-económico

En Junio del 2017 se publicó un estudio donde se expuso que la demografía de los diferentes grupos de la población y se previó que cambiará en los próximos años, titulado *The 2017 Revision of World Population Prospects* [13]. En él se estimó que la población mundial asciende a 7,6 mil millones, lo que implica que el mundo ha acogido a mil millones de habitantes con respecto a los últimos doce años.

[1]Telemetría: Conjunto de técnicas utilizadas para la medición de magnitudes físicas.

Tal como se refleja en la figura 2.1 la población continuará creciendo estimando que para el 2030 aumente la población a 8,6 mil millones y a 112,2 mil millones en 2100.

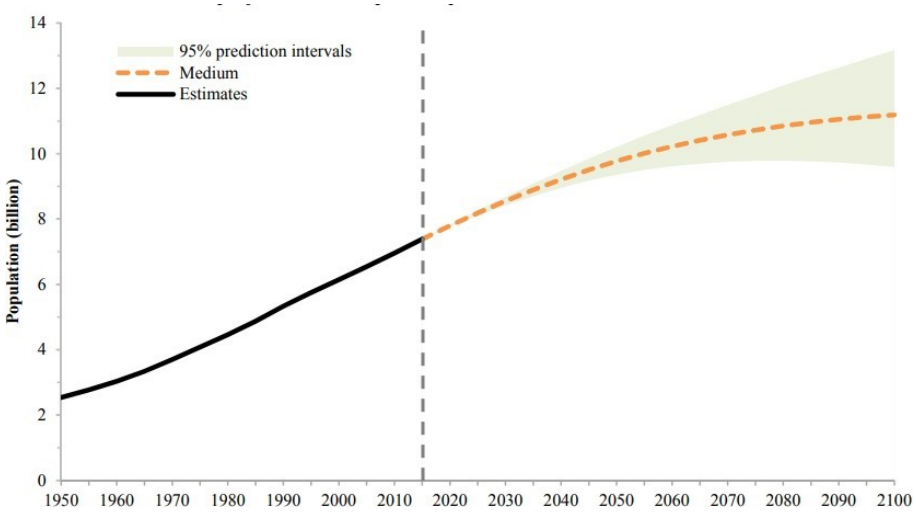

Figura 2.1: Estimación media de la población entre 1950-2100
Fuente: *Department of Economic and Social Affairs, Population Division (2017). World Population Prospects: The 2017 Revision. New York: United Nations.* [13]

Por otro lado, a nivel mundial el grupo de población que está creciendo con mayor rapidez es el de mayor de 60 años. Debido principalmente a la disminución de nacimientos y al aumento de la esperanza de vida. En 2017 este grupo representaba el 13 % de la población y crece cada año un 3 % [13].

En Europa, este grupo representa más del 25 % y se prevee que para dentro de unos 32 años representará la cuarta parte de la sociedad en todos los territorios con la excepción de África. Por tanto, la población está envejeciendo y esto amenaza a la mayoría de los sectores que sostiene nuestra sociedad y forma de vida, como el trabajo, las casas, la seguridad, las relaciones o las interacciones con la sociedad, familia o amigos... [14]

En la mayoría de los países del primer mundo, la mayoría de la atención a las personas mayores o con discapacidad está financiada por subvenciones del gobierno, pero con el envejecimiento de la sociedad, aumenta este coste. Por tanto, es necesario buscar soluciones integradas con la tecnología que disminuyan el coste [11].

2.3. Ayuda a la independencia

Es evidente que las personas mayores son más propensas a tener problemas de salud y a los efectos desfavorables de enfermedades repentinas o caídas, que pueden prevenirse o calmarse en cierta medida con sistemas de alarma y monitorización

en tiempo real [11].

Algunas de las soluciones y aplicaciones de IoT de última generación que se pueden utilizar en la domótica para la atención de personas mayores y discapacitados son [11]:

- **Asistencia de emergencia y respuesta:** La base de los sistemas de asistencia de emergencia y respuesta es un transceptor inalámbrico con un botón que el sujeto debe presionar en caso de que necesite ayuda. Al activar este dispositivo se envía una señal y se inicia la comunicación del usuario con el centro de atención asociado. Si se evalúa que el usuario necesita ayuda, se envían los servicios de emergencia necesarios. Una evolución de este sistema, proporciona estas alertas automáticamente cuando se recibe algún cambio significativo e inesperado en alguno de los signos vitales que se tiene monitorizados. Además permite que los miembros autorizados puedan acceder a los datos recopilados por el sistema. Para recoger estos datos, es necesario que el sujeto lleve consigo diferentes dispositivos de medición en función de los datos que se quieren recoger. Una tercera generación de este sistema, trataría de poner en contacto a los sujetos con otras personas en su misma situación a través de Internet con el objetivo de lidiar con el aislamiento, la soledad o la depresión.

- **Memoria:** Los sistemas de aviso automático permiten recordar a los sujetos que se tomen sus medicamentos, sus citas con el médico, que paseen al perro u otras tareas cotidianas. Estos mensajes de aviso se pueden programar en un dispositivo móvil o un reloj con mensaje de texto de manera económica y sencilla. Por otro lado, los dispensadores de medicamentos permiten que los sujetos dispongan de los medicamentos necesarios en el momento adecuado. Esto se puede combinar con los dispositivos de medición de las constantes vitales, de manera que si se pronostica algún cambio peligroso en los signos vitales, se proporcionen los medicamentos necesarios para solucionar estos problemas.

- **Vista y audición:** Las alarmas o sensores en puertas, timbres, cocina o detectores de humo alertan a las personas con problemas auditivos o visuales a detectar algunas de las amenazas que pueden surgir en las casas.

- **Supervisión de vídeo:** La supervisión de vídeo permite que los cuidadores tengan visión de lo que está sucediendo en las casas, para proporcionar la ayuda necesaria en cada caso.

2.4. Marco Regulador

En los últimos años se ha demostrado que la aplicación de algoritmos y técnicas sobre grandes conjuntos de datos, tiene grandes utilidades, tanto en sectores públicos como privados. Con la aplicación de distintos algoritmos, se determina si los clientes de algunos bancos, son candidatos seguros para que se les conceda un

préstamo, o si los estudiantes deben ser admitidos en la universidad. En ocasiones, estos algoritmos o técnicas son opacos o difíciles de analizar y dado que su aplicación va en aumento es importante que se fijen y regulen los límites, para garantizar que las decisiones que se toman con la información que ofrecen estos sistema sean realmente justas y precisas [15].

2.4.1. Legislación

La entrada en vigor del Reglamento General de Protección de Datos de la UE (GDPR), el 15 de mayo de 2018, ha provocado cambios significativos en muchos aspectos de la recopilación y procesamiento de datos. Se considera que es el primer cambio en la regulación de la privacidad de datos del siglo XXI [16] y afecta principalmente a tres áreas, el procesamiento de datos y perfiles, el derecho de una explicación y la responsabilidad:

- **Procesamiento de datos y perfiles**. A nivel de procesamiento de datos se imponen diferentes reglas que permiten a las organizaciones procesar datos personales con fines comerciales específicos, cumpliendo con las normas nacionales. Pero no permiten que se usen los datos para otros fines sin un permiso adicional del consumidor. Estas normas no aplican a la información anónima [17]. Por otro lado, como los consumidores tienen derecho al olvido, la información no debe ser guardada más que el tiempo necesario [18].

- **El derecho a una explicación**. En función del tipo de información se establece que los consumidores tienen derecho a que las decisiones no estén basadas únicamente en el procesamiento automatizado y que dicho procesado de su información personal sea transparente [18]. Esta norma ha creado una gran controversia, porque no precisa el alcance de las decisiones cubiertas en esta sección. En general, se prevé que tendrá un gran impacto en la contratación, solicitudes o decisiones de la concesión de servicios como los seguros [17].

- **Responsabilidad**. En este sentido, se establece que el controlador de la información es el responsable de que se cumplan las leyes establecidas [18]. Por tanto, las organizaciones que utilizan las decisiones automatizadas deben asegurarse de que el procesamiento de la información es justo y transparente, de que los procedimientos matemáticos y estadísticos son los apropiados y de establecer medidas que garanticen el resto de leyes establecidas [17].

2.4.2. Propiedad intelectual

En esta sección se especifican las licencias de las herramientas y de los conjuntos de datos que se han utilizado en este Trabajo Fin de Grado; las cuáles se detallan en las secciones 3.7 4.1 y 5.1.

Herramientas

- **RapidMiner Studio**, proporciona una licencia para el sector educativo que permite procesar filas de datos ilimitadas en un único procesador lógico.
- **Anaconda Distribution**, es una distribución de software libre y de código abierto de los lenguajes Python y R; que está orientada a simplificar la administración de paquetes, para algunas librerías, como: Pandas, Scikit-learn o Numpy.
- **Python 2.7**, es un lenguaje de código abierto propietario de la fundación *Python Software Foundation* (PSF), a partir de 2001. Su licencia se denomina *Python Software Foundation License* (PFSL) y es compatible con la licencia, GPL *(General Public License)*, que es la más extendida en código abierto.

  PFSL permite distribuir versiones modificadas sin obligar a que estos cambios sean de código abierto. Por tanto, es más permisiva que GPL, que pone especial atención en el *copyleft*, para que las versiones sean distribuidas con las mismas condiciones que las originales.
- **Pandas, Scikit-learn y Numpy** son librerías de código abierto que tienen licencia BSD *(Berkeley Software Distribution)*, que al igual que PSF es más permisiva que GPL y permite el uso del código en software privativo.

conjuntos de datos

Se han utilizado dos conjunto de datos, el primero de la competición *Senior Data Science: Safe Aging with SPHERE* [1] de la plataforma DrivenData [2]. En esta competición colaboran médicos, trabajadores sociales y científicos de universidades reconocidas como *Bristol*, *Reading* y *Southampton*. Dicha comunidad, basa su experiencia en los grupos líderes de Reino Unido de comunicaciones, visión artificial, cibernética, ciencia de datos y en dos corporaciones de reputación mundial en investigación y desarrollo, que son IBM y Toshiba. Tiene asociada la siguiente página web, donde se describen los resultados y reglas de la competición, además de los detalles del conjunto de datos de interés.

`https://irc-sphere.ac.uk/sphere-challenge/home`

El segundo conjunto de datos, *Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set* [19] es proporcionado por el repositorio de *machine learning*, UCI. La licencia de este conjunto permite la distribución de la misma sin modificaciones y prohíbe la distribución comercial o indebida [20].

2.5. Datos de interés

DrivenData [2] es una plataforma que presenta competiciones y desafíos de vanguardia, basados en el estudio de modelos estadísticos y predictivos. Los datos que proporcionan son abiertos y pertenecen a organizaciones no lucrativas.

La comunidad SPHERE ha creado sensores de bajo coste, escalables y de forma pasiva, que no requieren acción por parte del usuario y detectan problemas de salud, de forma simultánea, a través de datos y reconocimiento de patrones en un entorno no médico como el hogar. Con el desarrollo de esta tecnología, se pretende detectar caídas, un consumo de la medicación inadecuado y periodos de depresión o ansiedad.

Por otro lado, UCI (Machine Learning Repository) [3], es un repositorio que proporciona una colección de conjuntos de datos, usados para el análisis empírico de los algoritmos de aprendizaje automático.

3 | Ciencia de datos

La ciencia de datos ha adquirido un gran interés en los últimos años, debido a la amplia disponibilidad de grandes cantidades de datos, a la que se tiene acceso y a la necesidad de crear conocimiento útil a partir de toda esta nueva información. A lo largo del tiempo, se han desarrollado diversos algoritmos, métodos de aprendizaje y técnicas de limpieza o pre-procesado de datos, con el fin de conseguir información y conocimiento que va desde el análisis de mercado y detección de fraude hasta producción y exploración científica [21].

En este capítulo se describe el desarrollo de la ciencia de datos, las diferentes metodologías y los modelos utilizados para predecir las probabilidades de cada actividad (tabla 4.1, página 26) del conjunto de datos proporcionado por la competición *Senior Data Science: Safe Aging with SPHERE* [2] de la plataforma DrivenData.

3.1. ¿Qué es la ciencia de datos?

La ciencia de datos o *Data Science*, que es la terminología moderna para el llamado *Data Mining*, se define como el proceso de extracción de conocimiento de los datos (*knowledge mining from data*). Hay otros términos que también definen un significado similar como, el análisis de datos o patrones o la arqueología de datos. Otra alternativa es considerar la ciencia de datos como un sinónimo de otro término popularmente utilizado, *Knowledge Discovery from Data* o *KDD*, que en español equivale a "Descubrimiento del conocimiento de los datos" o como el conjunto de pasos esenciales para descubrir el conocimiento, siguiendo una metodología. El análisis predictivo permite reconocer tendencias y patrones a partir de conjuntos de datos. No es una ciencia absoluta, por tanto, el resultado de un modelo de predicción siempre contendrá un cierto error. En la siguiente sección se describen y comparan las metodologías más populares, para llevar a cabo dicho procedimiento [21].

3.2. Metodología

Las diferentes metodologías permiten llevar a cabo de forma lógica la construcción e implementación de proyectos *Data Science* en un entorno real; las más populares son SEMMA (*Sample, Explore, Modify, Model, Assess*) y CRISP-DM

(*CRoss-Industry Standard Process for Data Mining*) [22]. La figura 3.1, representa una encuesta del 2014 y 2007 donde se observa la popularidad de las diferentes metodologías.

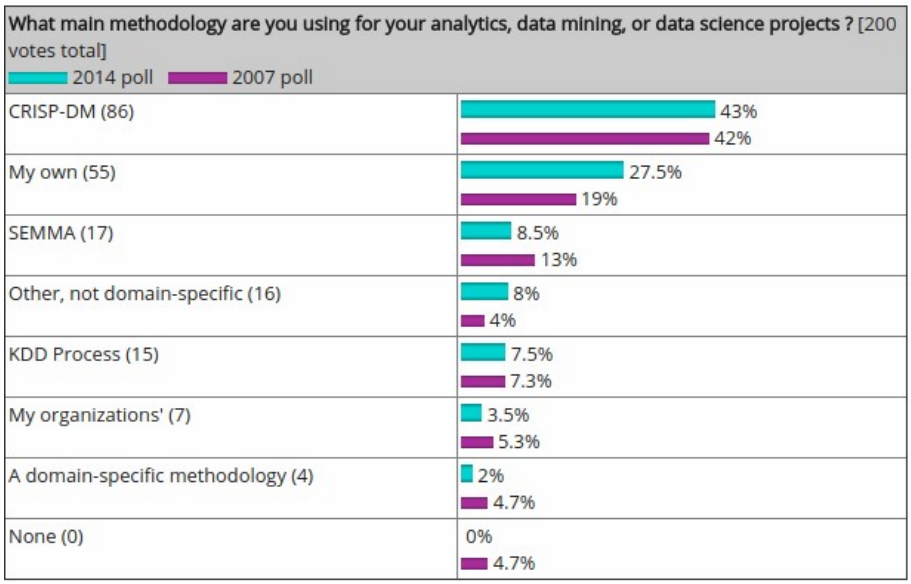

Figura 3.1: Metodologías de *Data Science*
Fuente: *KDnuggets* [23].

La metodología SEMMA, fue propuesta por *SAS Institute*, que es uno de los principales fabricantes de inteligencia empresarial software y consta de cinco fases: muestrear, explorar, modelar y evaluar. Por tanto, se empieza estudiando una pequeña parte de los datos, a continuación se examina todo el conjunto y se modifica y trata, para que en la siguiente fase se puedan aplicar los modelos convenientes. Finalmente, en la última fase se evalúan los resultados. Esta metodología solo tiene en cuenta las partes estadísticas, de modelado y de manipulación del proceso de *Data Mining*, excluyendo fases de análisis, diseño e implementación.

CRISP-DM, fue desarrollada por el consorcio de empresas de *data mining* y financiado por la Comisión Europea. Es un proceso de seis fases cíclico, donde se pueden usar varias iteraciones para permitir un resultado ajustado a los objetivos. En la primera fase, *Business Understanding*, se determina el objetivo a alcanzar. Posteriormente, en la segunda, *Data Understanding*, se extraen y seleccionan los datos y en la tercera, *Data Preparation*, se preparan los datos para aplicar el modelo (si los datos no tienen el formato correcto a la ahora de aplicar el modelo, se debe volver a esta fase, para corregirlos), mediante el pre-procesado y la selección de características. En la fase, *Modeling* se construye un modelo de aprendizaje, que se evalúa en la siguiente fase, *Evaluation*. Si el modelo no tiene un rendimiento o resultado suficientemente bueno se define una nueva iteración. De lo contrario, se pasa a la siguiente fase, *Deployment*, donde se procede al despliegue final del modelo. Dependiendo de los requisitos, esta fase puede ser simple o compleja, como generar un informe o implementar una repetición de la evaluación de los datos [24]. Esta metodología implica que los datos deben estar disponibles y ser válidos,

por tanto no se pueden recopilar datos nuevos. Tras la comparación de ambas metodologías, se concluye que la metodología CRISP-DM, es más completa porque se tiene en cuenta el objetivo de negocio a lograr, y por ello es la que se utiliza en este Trabajo Fin de Grado.

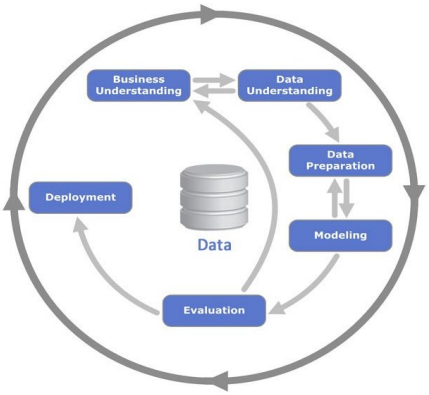

Figura 3.2: Fases de CRISP-DM
Fuente: *Wikipedia*

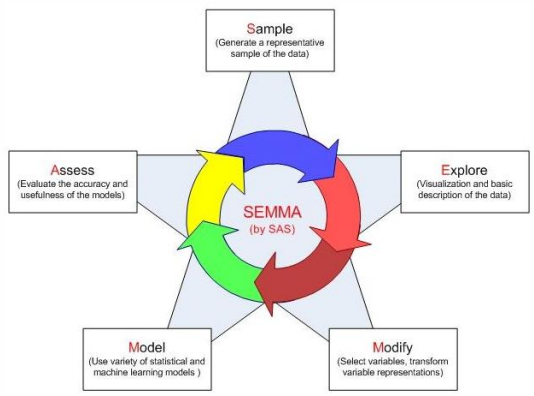

Figura 3.3: Fases de SEMMA
Fuente: *Gmelli*

3.3. Tipos de patrones de la ciencia de datos

En general, las tareas de la ciencia de datos se pueden clasificar en tres categorías: descriptiva, predictiva y prescriptiva [21]. Ninguna de las tareas es más importante que la otra, todas ellas coexisten y se complementan entre sí.

- Las **tareas descriptivas**, son aquellas que describen o resumen los datos y los convierten en algo interpretable por los humanos. Los análisis descriptivos son útiles porque permiten aprender de los comportamientos pasados y entender como influyen en resultados futuros.
- Las **tareas predictivas**, permiten extraer información de los datos para hacer predicciones. El análisis predictivo proporciona estimaciones sobre la probabilidad de un resultado futuro, teniendo en cuenta que en cualquier predicción siempre existe una cierta probabilidad de error.
- Las **tareas prescriptivas**, proporcionan una serie de consejos para llegar a un resultado concreto. El análisis prescriptivo cuantifica el efecto de la decisiones futuras con el objetivo de recomendar sobre posibles resultados, antes de tomar una decisión.

A lo largo de este Trabajo Fin de Grado se describirán algunas tareas descriptivas (secciones 4.1 y 5.1) y otras predictivas durante el desarrollo de los análisis predictivos (secciones 4.2 y 5.2).

3.4. Aprendizaje automático

Una parte importante de la ciencia de datos es el aprendizaje automático, que nació con el desarrollo de la inteligencia artificial (IA) y se promovió tras la publicación del paper *"Computing Machinery"* de *Turing* y su pregunta *"Can machines think?"* [25] que en español significa *"¿Pueden pensar las máquinas?"*.

El aprendizaje que una máquina realiza por sí misma mediante el uso de los datos y de los algoritmos se denomina, aprendizaje automático, y tiene como objetivo descubrir patrones y relaciones de los datos a través de un modelo que más tarde se utiliza para predicciones futuras [26].

Existen muchos tipos de algoritmos de aprendizaje automático, según su forma de aprendizaje, se pueden diferenciar cuatro tipos: aprendizaje supervisado, aprendizaje no supervisado y aprendizaje semi-supervisado.

- En el **aprendizaje supervisado** es necesario que cada instancia de datos tenga una variable etiquetada como valor de salida deseado, de acuerdo con el resto de variables o valores de entrada [26]. Algunos de los modelos más comunes que utilizan el aprendizaje supervisado son: arboles de decisión, redes neuronales artificiales, estimadores kernel, clasificadores naïve Bayes, *k-NN*...

- En el **aprendizaje no supervisado** a diferencia del supervisado no tiene ninguna etiqueta de salida. Por tanto, su objetivo es extraer conclusiones a partir de un conjunto de ejemplos sin etiquetar. Los algoritmos de aprendizaje no supervisado principales son: *clustering* y *principal component analysis* [26].

  - Los algoritmos de *clustering* utilizan algunas metodologías como *k-means* o agrupación jerárquica para agrupar los datos.
  - PCA o *principal component analysis*, se usa para la reducir el número de variables de modo que el rendimiento de aprendizaje se maximice, acelerando el entrenamiento y reduciendo la carga computacional (apartado 4.3.2).

- En el **aprendizaje semi-supervisado**, se utilizan tanto datos etiquetados como datos no etiquetados. Normalmente el conjunto de datos suele estar formado en su mayoría de datos no etiquetados y tienen una pequeña cantidad de datos etiquetados. Este tipo de aprendizaje también se suele definir como la combinación de usar algoritmos de aprendizaje supervisado y no supervisado, de manera que se combinan la ventajas de ambos.

3.5. Modelos de clasificación

En las secciones 4.2 y 5.2 se describen los experimentos realizados con diferentes modelos con el objetivo de determinar aquel que predice mejor las distintas actividades realizadas por el sujeto para cada conjunto de datos. En concreto se utilizan los siguientes tipos y modelos:

- **Lazy Learners:** Este tipo de clasificadores se denominan *"perezosos"* porque esperan a tener todos los datos antes de construir el modelo para un dato de *test* determinado; además a diferencia de los denominados *"ansiosos"* (como los árboles de decisión, los bayesianos...) cuando el modelo recibe un dato *train* lo almacena (o le hace un pequeño procesamiento) y espera a que un dato *test* generalice para clasificarlo en función de la similitud almacenada. Por tanto, se hace menos esfuerzo cuando se presenta una dato *train* que al hacer una clasificación o predicción para un dato *test* [21].

  - **K-Nearest Neighbours:** El *k-NN* se basa en el aprendizaje por analogía, es decir, por relación de semejanza entre los datos *test* con los datos *train* almacenados. En este tipo de clasificación se busca el espacio de patrón para los datos *train* que están más cerca de los datos *test* [27]. La "cercanía" se define en términos de distancia, como la distancia Euclídea o la distancia Manhattan, entre otras, de la siguiente manera:

    - **Distancia Euclídea:**

$$Dist_{xy} = \sqrt{\sum_{k=1}^{m}(x_{ik} - y_{jk})^2} \tag{3.1}$$

    - **Distancia Manhattan:**

$$Dist_{xy} = max_k|x_{ik} - y_{jk}| \tag{3.2}$$

- **Bayesian:** Los clasificadores bayesianos se basan en el *Teorema de Bayes* y predicen las probabilidades de que un dato pertenezca a una clase en particular. Se diferencian principalmente dos tipos:

  - **Naive Bayes:** Este tipo es comparable al rendimiento que tienen los árboles de decisión y los clasificadores de redes neuronales seleccionados. Se denominan los clasificadores bayesianos "ingenuos", porque suponen independencia condicional de clase, es decir, suponen que el efecto de un valor de un atributo en una clase es independiente de los otros atributos [21].

- **Bayesian Belief Networks:** Las redes credenciales bayesianas permiten la representación de dependencias entre subconjuntos de atributos [21].

Nota:

El *Teorema de Bayes*, se define como la probabilidad de que un dato X pertenezca a la clase C, dado que se conoce la descripción del atributo de X (fórmula 3.3).

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \tag{3.3}$$

- **Decision Tree:** Los árboles de decisión se consideran métodos no paramétricos, porque no presuponen ni el volumen de clase, ni la estructura del árbol, ni el crecimiento del árbol, a priori. Son métodos intuitivos, con posible representación visual.

  La estructura general de los árboles de decisión consta de un nodo raíz, de nodos internos, de hojas (o nodos finales) y de ramas. El nodo raíz se considera el punto inicial del árbol y representa uno de los atributos de entrada, los nodos intermedios son el resto de atributos de entrada. Las ramas representan los posibles valores de los atributos y las hojas o nodos finales los valores de las clases o etiquetas. En algunas ocasiones las hojas no representan etiquetas, porque algunos modelos de árbol no consiguen clasificar algunas hojas; en estos casos las hojas se marcan con una interrogación [26].

  Para elegir los atributos y el orden en el qué aparecen en el árbol se utilizan funciones de evaluación como la ganancia de información o el índice *gini*. Este último da lugar a árboles binarios, en cambio, la ganancia de información permite divisiones de los nodos en múltiples caminos y elige los atributos según el que tenga mayor ganancia de información al hacer la ramificación, es decir, la división de los nodos se hace según se estima que se va obtener una mejor clasificación final [21].

  Al construir un árbol de decisión algunas ramas pueden reflejar anomalías en los datos *train* por el ruido o valores atípicos. Para abordar este problema se utilizan los métodos de poda de árboles (*Tree pruning methods*), que eliminan las ramas que consideren que representan dichas anomalías. Los árboles podados tienden a ser más pequeños y menos complejos. Por lo general, son más fáciles de comprender, más rápidos y mejores para clasificar adecuadamente los datos *test* independientes.

  - **Rep Tree:** Es una variación de los árboles de decisión que permite construir dichos árboles más rápidamente utilizando la ganancia o varianza de información y podando los errores.

- **Ensemble Tree:** Los métodos de aprendizaje en conjunto (*Ensemble Learning*), utilizan múltiples algoritmos de aprendizaje de forma repetida para mejorar la precisión de clasificación de un algoritmo básico. Cada aplicación del algoritmo es una generación que se agrega al conjunto. Para clasificar un nuevo dato, cada generación clasifica inicialmente el caso independientemente de los demás y después se deriva a una única clasificación final teniendo en cuenta todos los resultados [28] [29]. Ademas de la técnica de la poda de árboles para mejorar la precisión de los árboles se utilizan otros métodos que combinan múltiples modelos, como el *"bagging"* o el *"boosting"*. Con el método ***"boosting"***, se predice la clase según una votación ponderada teniendo en cuenta los resultados de todos los árboles y dando un peso adicional a los puntos incorrectamente pronosticados por árboles anteriores. Con el método ***"bagging"***, se predice según los resultados con mayoría simple, pero en este caso, los distintos árboles no dependen de los resultados que hayan obtenidos los anteriores, por tanto son independientes entre ellos y utilizan una muestra de arranque del conjunto diferente para cada árbol [21] [30]. Dos métodos muy populares de aprendizaje en conjunto con árboles de decisión, son *Random Forest* y *Gradient Boosted Trees*.

  - **Random Forest:** Es un método propuesto por *Breiman* en 2001, cuyo fundamento es agregar una capa adicional de aleatoriedad de *bagging*. Además de construir cada árbol usando una muestra de inicio diferente de los datos, los bosques aleatorios cambian la forma en que se construyen los árboles. En árboles estándar como en el método *Decision Tree*, cada nodo se divide utilizando la mejor división entre todas las variables. En cambio, con *Random Forest* cada nodo se divide utilizando el mejor de un subconjunto de predictores elegidos al azar en ese nodo [30].
  - **Gradient Boosting Tree:** Es un modelo robusto, iterativo e interpretable con una precisión comparable al de otros métodos de aprendizaje estadístico complejo, como las redes neuronales. Tanto el procesado previo necesario y el ajuste de los parámetros es sencillo. Algunas de las mejoras que se ha hecho sobre este modelo fue la de *Friedman* en 2001, con la incorporación de regularización para evitar el sobreajuste y en 2002 con la introducción de aleatoriedad en el proceso de adaptación del modelo [31].

- **Rules:** En los clasificadores basados en reglas el modelo representa un conjunto de condiciones IF-THEN. Estas reglas se pueden generar a partir de un árbol o directamente desde los datos *train* utilizando un algoritmo de cobertura secuencial o algoritmos de clasificación asociativa. Una regla IF-THEN forma una expresión de la siguiente forma:

  IF condition THEN conclusion

  - **Rule Induction:** Los algoritmos secuenciales son el enfoque más utilizado para extraer conjuntos de reglas de clasificación. Este

aprendizaje secuencial de las reglas está en contraste con la inducción del árbol de decisión. El camino a cada hoja en un árbol de decisión corresponde a una regla, de esta manera se puede considerar la inducción del árbol de decisión como el aprendizaje de un conjunto de reglas simultáneamente.

- **Linear Models:** Los clasificadores basados en árboles de decisión o reglas funcionan de manera más natural con atributos nominales. En cambio, para algunos atributos numéricos existen diferentes modelos lineales que clasifican de manera más natural y eficiente [32]. Algunos de los más simples son: *Linear Regression* o *Logistic Regression*.

  - **Linear Regression:** La regresión lineal se puede usar para clasificación, de manera que se realiza una regresión para cada clase, estableciendo como resultado 1 a las instancias que pertenezcan a la clase y cero a las que no pertenezcan a la clase. Sin embargo, este modelo tiene algunos inconvenientes como que los resultados no son siempre probabilidades adecuadas dentro del rango de 0 a 1 y que los errores no son estadísticamente independientes [32].
  - **Logistic Regression:** La regresión logística construye los modelos lineales basándose en una variable objetivo transformada; por tanto, no aproxima directamente los valores de 0 a 1 a diferencia de la regresión lineal.
    - **Simple Logistics:** es una variación del *Logistic Regression* que utiliza el algoritmo *boosting* para construir el modelo.

- **Artificial Neural Networks:** Las redes neuronales artificiales (ANN), proporcionan un método general y práctico para el aprendizaje de ejemplos de valores reales, valores discretos y valores de vectores. Su aprendizaje es robusto a los errores en los datos de entrenamiento, por tanto es adecuando para datos de sensores complejos y ruidosos. Las redes más populares son el *Perceptron* y *Deep Learning* [32].

  - **Perceptron:** El modelo de perceptrón que propuso *Frank Rosenblatt* en 1958, estaba compuesto por una neurona con un vector de entradas $\bar{x}$ de valores reales. Con estas entradas se calcula una combinación lineal y se emite un 1 si el resultado es mayor que algún umbral y -1 en caso contrario.

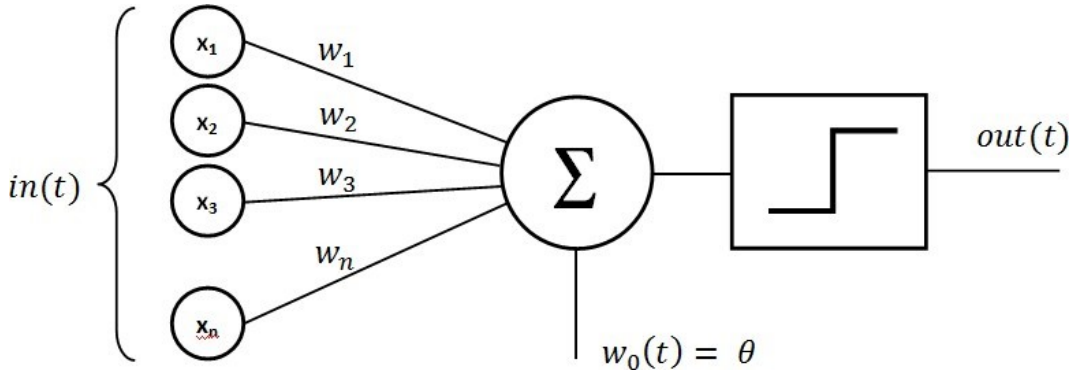

Figura 3.4: Perceptrón
Fuente: *Wikimedia Commons*

En la ecuación 3.4 se observa que para un vector de entradas $\bar{x}$ la salida calculada por el perceptron es:

$$o(x_1, ...x_n) = \begin{cases} +1 & w_0x_0 + w_1x_1 + ... + w_nx_n > 0 \\ -1 & cualquier\ otro\ valor \end{cases} \tag{3.4}$$

donde cada w_i representa un peso que determina la contribución de la entrada x_i a la salida del perceptrón [33]. De esta manera, para que el perceptrón produzca un 1 la combinación lineal de las entradas $(w_1x_1 + ... + w_nx_n)$ debe superar al peso $(-w_0)$. Para simplificar la notación de la ecuación 3.4, se suele suponer que $x_0 = 1$. Otra forma alternativa de representar la ecuación 3.4 es:

$$o_(\bar{x}) = sgn(\bar{w} \cdot \bar{o}) \tag{3.5}$$

donde

$$sgn(y) = \begin{cases} +1 & y > 0 \\ -1 & otherwise \end{cases} \tag{3.6}$$

Para determinar el vector de pesos, $\bar{w}$, que hace que el perceptrón produzca la salida de ± 1 correcta, para cada uno de los datos *train*, se utilizan algunos algoritmos como, la regla del perceptrón y la regla delta. Ambos convergen en hipótesis aceptables pero diferentes.

Una forma de determinar un vector de peso aceptable es comenzar con pesos aleatorios y luego aplicar iterativamente el perceptrón a cada ejemplo *train*, modificando los pesos del perceptrón siempre que clasifique erróneamente uno. Este proceso se repite, tantas veces como sea necesario hasta que el perceptrón clasifique correctamente todos los ejemplos de entrenamiento. Los pesos se modifican en cada paso de acuerdo con la regla del perceptrón (ecuaciones 3.7 y 3.8), que modifica el peso w_i asociado con la entrada x_i.

$$w_i \leftarrow w_i + \Delta w_i \tag{3.7}$$

donde

$$\Delta w_i = \eta(t - o)x_i \tag{3.8}$$

En la ecuación 3.8 t representa la salida objetivo para el ejemplo *train* analizado y η representa la velocidad de aprendizaje, que es una constante positiva, cuyo objetivo es moderar el grado en que se cambian los pesos en cada paso. Normalmente se suele establecer un valor pequeño como 0 o 1, durante el aprendizaje este valor se degrada al aumentar la cantidad de iteraciones de ajuste de peso.

La regla del perceptron no funciona correctamente si los datos *train* no son separables linealmente. En cambio, con la regla delta si los datos *train* no son separables linealmente, el modelo converge hacia la aproximación que mejor se ajuste al objetivo. Esta regla usa el descenso por gradiente para buscar el espacio de hipótesis de posibles vectores de ponderación para encontrar los pesos que mejor se adapten a los datos train.

El descenso por gradiente se puede aplicar sobre cualquier conjunto de datos si el espacio de hipótesis contiene hipótesis continuamente parametrizadas y si el error puede diferenciarse con respecto a estos parámetros de las hipótesis. La principal dificultad de este algoritmo de optimización es encontrar la convergencia a un mínimo global o en su defecto a un mínimo local adecuado, porque este proceso requiera de demasiados pasos o porque si hay múltiples mínimos locales de error, no hay garantía de que el procedimiento encuentre el mejor [32].

- **Perceptrón Multicapa y Deep Learning:** Las redes de aprendizaje profundo (DL) o el perceptrón multicapa (MLP) son combinación de varias neuronas de manera que se forma un modelo que proporciona divisiones más complejas con las que se pueden obtener mejores resultados. Como se indica en la figura 3.5 las capas intermedias entre la capa de entrada y la de salida se denominan capas ocultas.

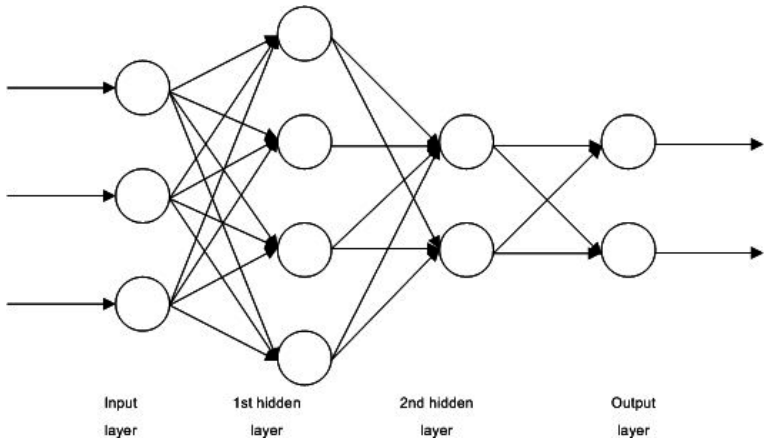

Figura 3.5: Perceptrón Multicapa
Fuente: *Wikimedia Commons*

Estos modelos se denominan *feedforward* porque la información fluye a través de la función que se evalúa y de los cálculos intermedios hasta la

salida. La redes con retroalimentación se denominan redes neuronales recurrentes (RNN).

Para generalizar los modelos, evitar el sobreajuste y reducir el error sobre los datos *test* se utilizan distintas técnicas de regularización como la regularización L1 o L2 [34]. Ambos actualizan la función de coste (F_c) agregando un término de regularización a la pérdida (P) como se muestra en las fórmulas 3.9 y 3.10.

Para L1:

$$F_c = P + \frac{\lambda}{2m} * \sum \|w\| \tag{3.9}$$

Para L2:

$$F_c = P + \frac{\lambda}{2m} * \sum \|w\|^2 \tag{3.10}$$

La regularización de L2 penaliza el valor cuadrado del peso, provocando que estos pesos se aproximen a cero (pero no exactamente cero). En cambio, con L1 se penaliza el valor absoluto de los pesos, por tanto se tiende a conducir a algunos pesos a exactamente cero. En general, con L2 se obtienen mejores resultados y L1 suele ser útil para comprimir el modelo [35] [36].

3.6. Criterios de evaluación de modelos

La determinación de un buen modelo depende de los intereses finales. Mediante los procesos de evaluación se puede verificar si un modelo es confiable para predicciones futuras o en el caso de querer comparar varios modelos, mediante estos procesos se puede averiguar y justificar el mejor modelo.

En esta sección se van a definir las diferentes medidas y métricas que se han utilizado para evaluar la precisión con la que cada modelo se adapta a los objetivos y estima el rendimiento global.

3.6.1. Métricas y medidas de evaluación del rendimiento

Antes de describir las diferentes medidas de evaluación, es necesario tener en cuenta características o métricas como la velocidad, la robustez, la escabilidad o interpretabilidad de los modelos [26].

- La velocidad, hace referencia al coste computacional durante la construcción y uso del modelo.
- La robustez, es el grado de fiabilidad de las predicciones cuando hay datos erróneos o *NaNs* (datos *missing*).

- La escabilidad, se evalúa como la precisión de las predicciones de los distintos clasificadores en grandes cantidades de datos.
- La interpretabilidad, se define como el grado en el que los resultados se pueden entender o interpretar, por tanto, se suele considerar que es subjetiva.

A pesar de la importancia de estas métricas, las medidas más utilizadas son las fundamentadas en métodos y técnicas de evaluación porque ofrecen resultados más realistas y fáciles de calcular, como las técnicas basadas en el *accuracy* o el cálculo del *brier score*.

El *accuracy* se calcula como el número de aciertos al relacionar las observaciones predichas y las observaciones reales. Para calcular las métricas basadas en el *accuracy* se utiliza la matriz de confusión, que es una matriz de mxm dimensiones, siendo m igual al número de clases o actividades. Esta matriz contiene información sobre las clasificaciones etiquetadas y las predichas [26] .

El *Brier Score*, es una función de calificación que mide la precisión de las predicciones probabilísticas, cuanto menor sea la puntuación medida, mejor será el rendimiento del algoritmo [37]. La mejor puntuación posible es 0, para una precisión total y la puntuación mas baja posible es 1, lo que significa que el pronóstico fue completamente inexacto [38]. Su formulación más común se define de la siguiente manera:

$$BS = \frac{1}{N}\sum_{t=1}^{N}\sum_{i=1}^{R} w_c(f_{t,i} - o_{t,i})^2 \tag{3.11}$$

En la ecuación 3.11, R es el número de clases posibles y N el número de datos *test*. Por otro lado, $f_{f,i}$ indica la probabilidad real, de que el dato pertenezca a una clase, $o_{f,i}$ marca la probabilidad que predice el algoritmo elegido y w_c es el peso para cada clase.

3.6.2. Evaluación de medidas

Para obtener un estimación confiables de medidas como el *accuracy* o el *Brier Score* se pueden utilizar algunas técnicas, tales como *holdout* o *k-fold cross validation* [21] [26].

- **Holdout**: En este método el conjunto de datos se divide en dos conjuntos independientes aleatoriamente, uno de ellos se considera el de *train*(o entrenamiento) y el otro el de *test* (o prueba). La porción de división utilizada más adelante en la sección 4.2 y 5.2 es: 80 % como conjunto *train* y el 20 % como *test*. El conjunto *train* se utiliza para construir el modelo y el *test* para evaluarlo, como se puede ver en la figura 3.6. Esta evaluación se considera pesimista porque solo una parte de los datos iniciales se usa para construir el modelo.

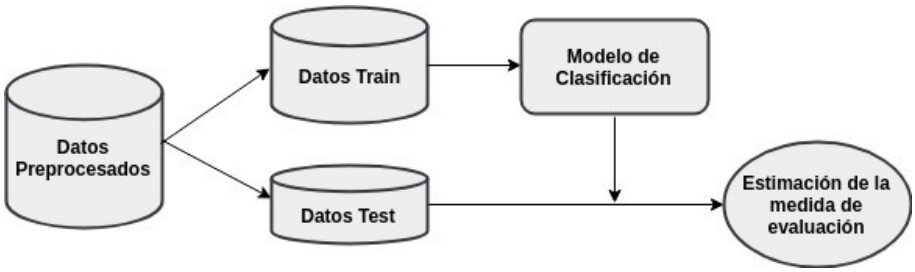

Figura 3.6: Técnica *holdout*

Este método *holdout* solo se suele usar cuando hay suficientes datos, de manera que un subconjunto se pueda usar para el entrenamiento y otro por separado para las pruebas. La aleatoriedad de este método puede provocar que ambos conjuntos no representen de manera proporcional las diferentes clases a predecir, por tanto, es necesario hacer esta comprobación y en caso necesario muestrar el conjunto de datos.

Dicho método se utiliza en la sección 4.2 y 5.2 para entrenar y evaluar los diferentes modelos.

- **Cross-validation**: En esta técnica los datos *train* se dividen aleatoriamente en k subconjuntos $D_1, D_2, ..., D_k$ mutuamente excluyentes de aproximadamente el mismo tamaño. En la iteración i, el subconjunto D_i se reserva como el conjunto de validación, y las particiones restantes se usan colectivamente para entrenar el modelo. Es decir, en la primera iteración ($i = 1$), los subconjuntos $D_2, ..., D_k$ se utilizan colectivamente como el conjunto de entrenamiento para obtener un primer modelo, que se valida con el subconjunto D_1; la segunda iteración ($i = 2$) se entrena con los subconjuntos $D_1, D_3, ..., D_k$ y se valida en D2; y así sucesivamente durante k veces porque cada subconjunto va a actuar de validación una vez (Figura 3.7).

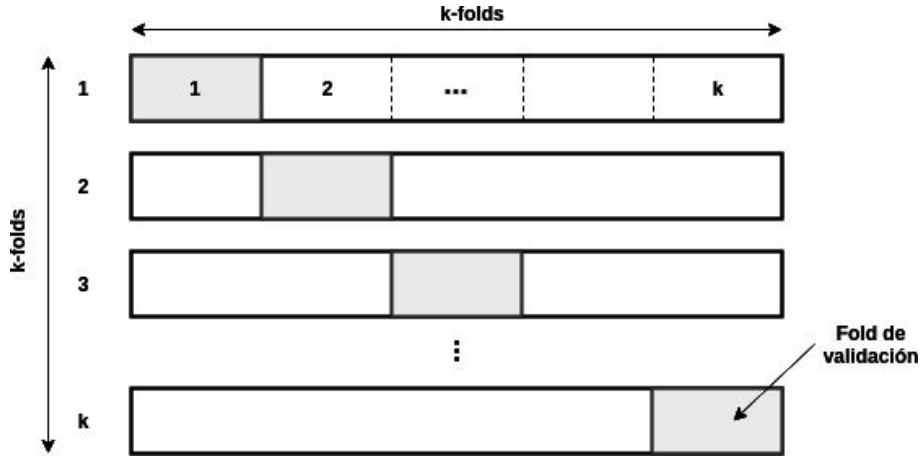

Figura 3.7: Validación cruzada *k-fold*

Cada muestra usa la misma cantidad de veces para el entrenamiento y una vez para la validación. Para validar cada conjunto de validación se utiliza

la medida del *accuracy*. Para al estimación del *accuracy* de la validación cruzada (CVA $\sim$ *Cross Validation Accuracy*), se calcula el promedio del *accuracy* calculado en cada *fold*, de la siguiente manera:

$$CVA = \frac{1}{k} \sum_{k}^{i=1} a_i \tag{3.12}$$

En la ecuación 3.12, k es el número de *folds* o subconjuntos y a_i es el *accuracy* de cada *fold*. Esta técnica tiene como objetivo minimizar el sesgo asociado con el muestreo aleatorio, por ello hace un uso completo de los datos.

En los apartados 4.2.2 y 4.2.3 se utiliza esta técnica para optimizar los algoritmos, de manera que se determinan los valores de los diferentes parámetros con los que se obtienen mejores resultados en la fase de entrenamiento.

3.7. Tecnologías utilizadas

Para implementar los experimentos se ha utilizado la herramienta *RapidMiner* y el lenguaje de programación Python 2.7.14 con la Suite Anaconda Distribution.

3.7.1. RapidMiner

RapidMiner es una herramienta de software para equipos de ciencia de datos que combina preparación de datos, aprendizaje automático y despliegue del modelo predictivo.

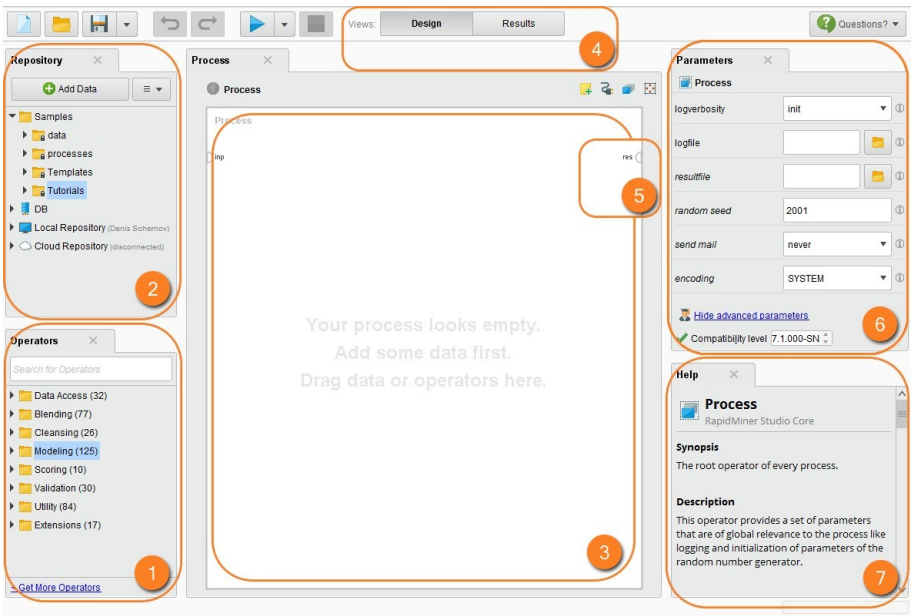

Figura 3.8: Vista de diseño de la pantalla principal de RapidMiner
Fuente: *RapidMiner*

1. **Operators (Operadores)**. En este submenú se encuentran los bloques principales de cualquier modelo en *RapidMiner*, los operadores. Estos bloques reciben una entrada, ejecutan un proceso, que puede ser un clasificador como el *Random Forest* o un filtro sobre los datos y devuelven el resultado del procesamiento. *RapidMiner* provee más de 1500 operadores distintos que se pueden buscar mediante la interfaz de navegación o mediante una búsqueda por nombre.

2. **Repository**. Es donde se almacena tanto los datos como los procesos en *RapidMiner*. Aunque es posible leer los datos directamente de un fichero localizado en una carpeta externa, *RapidMiner* es mucho más eficiente si se cargan los datos en su repositorio.

3. **Process (Procesos)**. El panel principal de procesos, es el área principal de trabajo sobre el que se van a construir nuestros modelos. Aquí es donde se van a arrastrar bloques y se va a interactuar con ellos.

4. **Views (Vistas)**. Aquí puedes acceder a distintas vistas principales con distintas funcionalidades. Por ejemplo puedes cambiar desde una vista del modelo en bloques a una vista de los resultados obtenidos en un *Dashboard*. Hay procesos que añaden nuevas vistas e incluso *RapidMiner* te proporciona una herramienta para crear tus propias vistas.

5. **Ports (Puertos)**. Son los puntos de anclaje mediante los cuales se produce el intercambio de datos entre bloques qué están representados siempre por unos círculos y una etiqueta que te indica que datos van a circular por ese puerto.

6. **Parameters (Parametros)**. En este panel se facilita la configuración de cada uno de los operadores, por ejemplo los hiperparámetros de un modelo de clasificación. *RapidMiner* dispone de dos clases de parámetros, los regulares y los normales, así como una funcionalidad de sugerencias inteligentes para los valores de cada uno de estos.

7. **Help (Ayuda)**. Se muestra un panel con información sobre el operador, no solo práctica sino también teórica mediante enlaces que amplían la información.

RapidMiner permite la integración con Weka [39] añadiendo la extensión *Weka Extension* al entorno de la herramienta para utilizar algunos modelos como:

- *W-SimpleLogistic*
- *W-REPTree*

Además esta herramienta cuenta con cientos de miles de usuarios activos, dato que nos puede proporcionar una idea de su utilidad. Entre sus principales ventajas frente a otros softwares es que es una plataforma unificada con un entorno de programación visual, lo que permite una rápida construcción y puesta a punto de modelos, de forma intuitiva y eficiente al estar basado en diagramas de bloque.

3.7.2. Lenguaje de programación Python

Python es un lenguaje de programación interpretado, orientado a objetos y de alto nivel con semántica dinámica. Su suite Anaconda Distribution ofrece una gran cantidad de funcionalidades y librerías para *Data Science* que permiten desarrollar aplicaciones y predecir variables de una manera más eficiente, rápida y sencilla.

Algunas de las librerías de Python utilizadas son:

- Pandas: es una librería que proporciona estructuras de datos y herramientas de análisis de datos.
- Numpy: es la librería fundamental en Python. Contiene, entre otras cosas: un potente objeto de *N-dimensional array* y capacidad para *random number* entre otras funcionalidades.
- Scikit-learn (sklearn): es una librería de aprendizaje máquina, que cuenta con varios algoritmos de clasificación, regresión y agrupación, incluyendo *support vector machines* y *random forests* entre otros.

La principal ventaja de este lenguaje interpretado, es que favorece la creación de códigos más legibles que otras opciones debido a su organización y sencillez. Sin embargo Python también posee otras ventajas funcionales como por ejemplo la del tipado dinámico, que permite cambiar de tipo de variable y proporciona mucha flexibilidad y potencia a la hora de realizar códigos complejos.

4 | Escenario 1

Tras completar la fase *Business Understanding* fijando como objetivo final, reconocer el movimiento de individuos a partir de los datos extraídos de un conjunto de sensores distribuidos en las estancias de una casa inteligente, en este capítulo se detallan las características del conjunto de datos proporcionado por la competición *Senior Data Science: Safe Aging with SPHERE* [1] y los cambios, análisis predictivo y otras técnicas aplicadas en cada experimento realizado.

4.1. Estudio de los datos

A lo largo de los siguientes apartados se desarrollan los atributos o variables que se han tratado en este conjunto de datos.

4.1.1. Descripción del conjunto de datos

El estudio y la comprensión de los datos corresponden a la fase *Data Understanding* de la metodología CRISP-DM (sección 3.2, página 10). En este conjunto de datos se distinguen un total de 20 actividades y cada una de ellas se define con una etiqueta. En la tabla 4.1 se especifica qué actividad representa cada etiqueta.

Nombre	**Actividad**	**Nombre**	**Actividad**
a_ascend	subir escaleras	*p_stand*	de pie
a_descend	bajar escaleras	*t_bend*	de pie inclinado
a_jump	saltar	*t_kneel_stand*	de rodillas a de pie
a_loadwalk	transportar algo	*t_lie_sit*	tumbado a sentado
a_walk	caminar	*t_sit_lie*	sentado a tumbado
p_bent	inclinado	*t_sit_stand*	sentado a de pie
p_kneel	de rodillas	*t_stand_kneel*	de pie a de rodillas
p_lie	tumbado	*t_stand_sit*	de pie a de rodillas
p_sit	sentado	*t_straighten*	de pie a sentado
p_squat	en cuclillas	*t_turn*	girarse

Tabla 4.1: Descripción de las 20 actividades

Todas las etiquetas tienen un prefijo: el prefijo *a_ (ambulation activity)* representa aquellas actividades de un solo movimiento continuo; el prefijo *p_ (static postures)* las posturas estáticas, es decir, momentos en los que los participantes están parados; y el prefijo *t_ (transition)* las transiciones entre posturas. Para la captura de estas actividades se utilizan tres tipos diferentes de sensores: acelerómetros, cámaras y sensores ambientales. Todos participantes llevan un dispositivo sujeto con una correa a la muñeca dominante, que tiene un acelerómetro triaxial que muestrea a 20 Hz. El dispositivo comunica los datos a diferentes receptores ubicados dentro de la casa, de forma inalámbrica y registra la potencia recibida en cada uno de los cuatro puntos de acceso en unidades de dBm.

Se dispone de tres cámaras de vídeo, una en la sala de estar, otra en el pasillo y otra en la cocina. Para proteger el anonimato de los participantes, los datos de vídeo en bruto no son proporcionados, y en su lugar se proporciona la información en coordenadas 2D y 3D. Por último, en cada habitación hay un sensor ambiental de presencia pasiva por infrarrojos (PIR).

El diseño que tiene la casa inteligente, es el de una casa independiente de dos habitaciones, un baño, un aseo, un recibidor, una cocina, un salón y un estudio, distribuidos en dos plantas, según se observa en la figura 4.1.

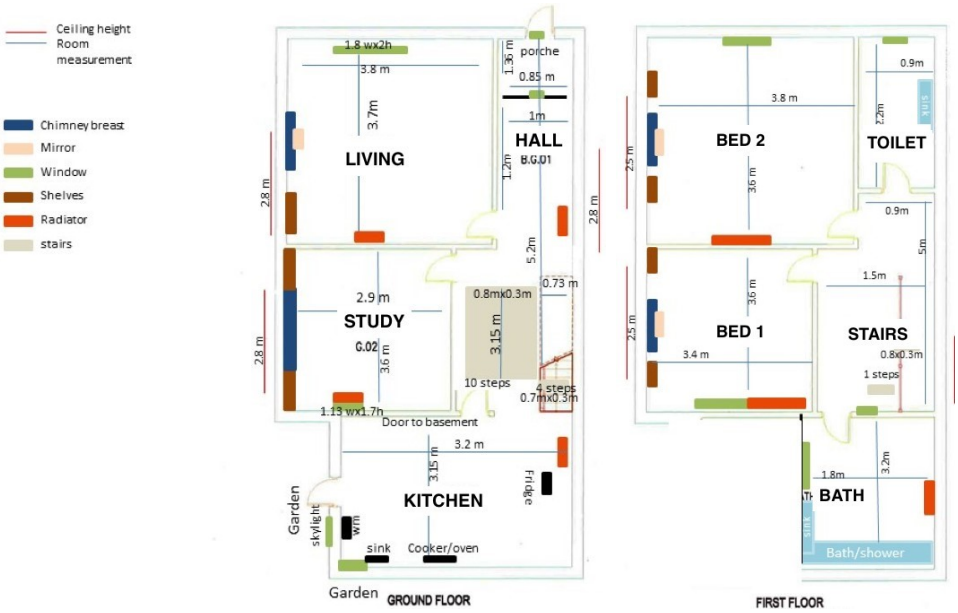

Figura 4.1: Plano de la casa
Fuente: *DrivenData*

Para etiquetar las muestras se escogió a un equipo de 12 profesionales que ayudados de una videocámara portátil grabaron toda la secuencia de actividades que los sujetos realizaban en la vivienda, en formato 4K y a 25 fotogramas. La sincronización entre el reloj y la cámara se logró enfocando con la cámara a un reloj digital sincronizado con el reloj del resto de sistemas de monitorización. Todos los datos grabados se recopilaron bajo código únicos, denominados secuencia de datos.

Para crear las etiquetas, se usó una herramienta llamada ELAN [40]. Esta herramienta se usa para la creación de anotaciones complejas en vídeos y audios; y fue desarrollada por el *Max Planck Institute* para la Psicolingüistica en Nijmengen, Holanda. Junto a las etiquetas también se realizaron otro tipo de anotaciones como la habitación en la que se encontraba el sujeto en ese preciso momento.

El conjunto de datos *train* está dividido en seis archivos principales de formato CSV, *targets.csv*, *pir.csv*, *acceleration.csv*, *video_hallway.csv*, *video_living_room.csv* y *video_kitchen.csv*; cada uno contiene una o varias columnas que especifican el tiempo en el que se capturó cada variable (tabla 4.2). En cambio, el conjunto de datos *test* que proporcionó la organización no contiene el archivo *targets.csv*, que tiene las variables objetivo. Por tanto, es necesario reservar una parte de los datos de entrenamiento, *train*, para la evaluación de los resultados.

targets.csv
[***start***, ***end***, *a_ascend, a_descend, a_jump, ..., t_straighten, t_turn*]
pir.csv
[***start***, ***end***, *name, index*]
acceleration.csv
[***t***, *x, y, z, Kitchen_AP, Loungue_AP, Upstairs_AP, Study_AP*]
videos_*.csv
[***t***, *centre_2d_x, centre_2d_y, ..., bb_3d_flt_x, bb_3d_flt_y, bb_3d_flt_z*]

Tabla 4.2: Columnas de los archivos
*(* = hallway / kitchen / living)*

El archivo *targets.csv*, que contiene las etiquetas para la clasificación, está formado por 22 columnas, las dos primeras son *start* y *end*, que representan la hora de inicio y de fin de las actividades de la tabla 4.1. Las 20 columnas restantes representan cada actividad con una probabilidad.

El archivo *pir.csv* contiene la hora de activación y desactivación de los sensores PIR de cada habitación. Tiene cuatro columnas, las dos primeras son *start* y *end*, que representan la hora de inicio y de fin de la activación del sensor. Las otras dos son *name* e *index*, que representan el nombre de sensor PIR y el índice al que corresponde dicho sensor, comenzando en 0, respectivamente (en la tabla 4.3 se observa la correspondencia de cada estancia con su índice). Por tanto, con los datos de este archivo obtenemos la habitación en la que está situado cada sujeto durante cada actividad.

index	*name*
0	*bath*
1	*bed1*
2	*bed2*
3	*hall*
4	*kitchen*

index	*name*
5	*living*
6	*stairs*
7	*study*
8	*toilet*

Tabla 4.3: Index correspondiente a cada estancia

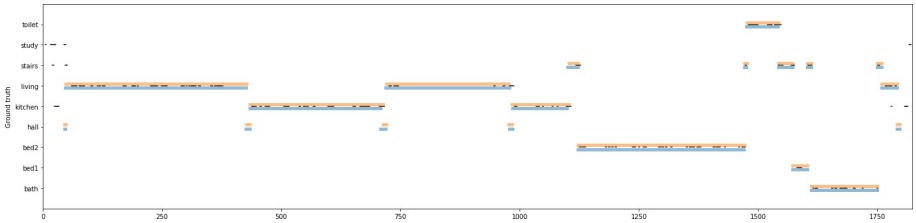

Figura 4.2: Ejemplo de los datos PIR de la secuencia de datos 00001 de *train*

En la figura 4.2 se muestra un ejemplo de registro de datos PIR de entrenamiento. El eje de la izquierda indica las diferentes estancias de la casa y las líneas horizontales señalan el tiempo en el que el sujeto permanece en cada estancia.

El archivo *accelerations.csv* contiene la información que recogen los puntos de acceso o receptores ubicados dentro de la casa, del dispositivo equipado con un acelerómetro triaxial que llevan en la muñeca los sujetos. Está dividido en ocho columnas: la primera es t e indica el momento en el que empieza la grabación con respecto al inicio de la secuencia. Las tres siguientes son x, y y z que tienen la información que proporciona el acelerómetro en cada eje. Por último, están las columnas *Kitchen_AP*, *Lounge_AP*, *Upstairs_AP* y *Study_AP* donde se especifica el indicador de intensidad de la señal recibida (RSSI) en los puntos de acceso de la cocina, del salón, del piso superior y del estudio respectivamente.

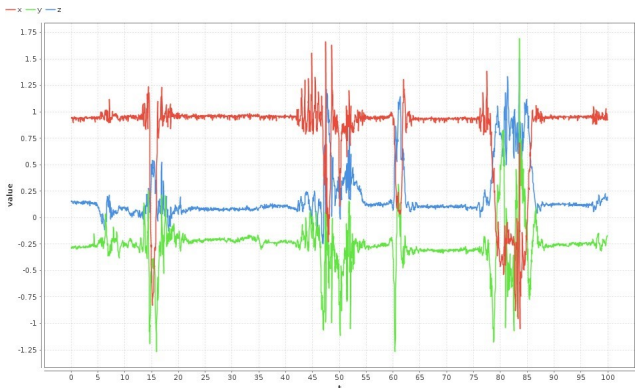

Figura 4.3: Ejemplo de los datos de (x, y, z) de *acceleration.csv* de train

En la figura 4.3 se representa un registro de datos *acceleration* de entrenamiento a lo largo del tiempo. El eje horizontal representa el tiempo y el vertical los diferentes valores de posición que registran los puntos de acceso.

Por último, los archivos *video_hallway.csv*, *video_kitchen.csv* y *video_living_room.csv* contienen las coordenadas 2D y 3D de las cámaras RGB-D del vestíbulo, la cocina y el salón respectivamente. Los tres están divididos en dieciséis columnas 4.2. La columna t indica la hora de la grabación con respecto al inicio de la secuencia, el resto de columnas contienen los valores de las coordenadas (x, y) o (x, y, z) del centro, la esquina inferior derecha y la esquina superior izquierda del 2D y 3D *bounding box*.

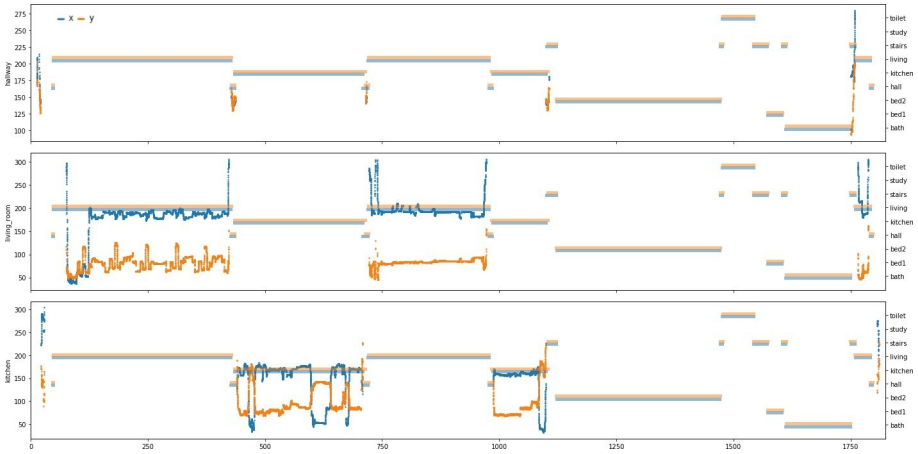

Figura 4.4: Ejemplo de las columnas *centre_2d* de los archivos de los vídeos de *train*

En la figura 4.4 se observan los datos que se recogen de la cámara RGB-D cuando el sujeto está en el salón o en la cocina. El eje horizontal representa el tiempo, el vertical de la izquierda los valores que recogen las cámaras y el de la derecha representa las diferentes estancias donde se puede situar el sujeto. De manera que si el sujeto se encuentra en el *living* solo debe recoger datos esta cámara, y sucesivamente.

4.1.2. Pre-procesado de Datos

Para descubrir patrones relevantes es necesario un volumen de datos suficientemente grande y de calidad. Con el objetivo de mejorar la calidad del conjunto de datos elegido y por tanto, mejorar los resultados finales se aplican diferentes tareas de pre-procesado de datos. Este proceso corresponde a la fase *Data Preparation* de la metodología CRISP-DM (sección 3.2, página 10), algunas de las tareas que se van a describir en este capítulo son:

- Integración de datos.
- Limpieza de datos, mediante la eliminación de datos desconocidos o erróneos.
- Eliminación o generación de atributos.
- Transformación de datos, como el suavizado o *smoothing*

Los archivos con formato CSV, *targets.csv*, *pir.csv*, *acceleration.csv*, *video_hallway.csv*, *video_living_room.csv* y *video_kitchen.csv* tienen diferentes períodos de tiempo. Para integrarlos y unificarlos, se crea la columna o atributo *t_round*, que especifica la marca temporal del conjunto de probabilidades de cada actividad en cada secuencia. Hay un total de diez secuencias y cada secuencia tiene estos seis tipos de archivos.

En el archivo *targets.csv* por cada marca temporal se indica un conjunto de probabilidades de cada actividad desde el inicio de la secuencia hasta el final (tabla 4.4), por tanto, la columna *start* se corresponde con la columna *t_round* creada.

t_round	***start***	***end***	***a_ascend***	***a_descend***	***a_jump***	***a_walk***	***...***
46	46	47	0,0	0,0	0,0	1,0	...
47	47	48	0,0	0,0	0,0	1,0	...
48	48	49	0,0	0,0	0,0	0,5	...
49	49	50	0,0	0,0	0,0	0,4	...

Tabla 4.4: Ejemplo del archivo *targets.csv* de la secuencia 00001 de *train*

En cambio en *acceleration.csv*, fue necesario redondear por unidades la columna *t* para que coincida con una de las marcas temporales de la columna *t_round* (tabla 4.5). Aproximar cada instante de tiempo *t*, a la marca temporal *t_round*, crea varios valores de aceleración por cada valor de marca temporal. Por tanto, fue necesario agrupar dichos valores de aceleración por su marca de tiempo, y mediante la función *groupby* se calculó la media y la desviación típica de cada grupo (tabla 4.6). De esta manera, solo hay un conjunto de valores de aceleración para cada conjunto de probabilidades anotadas (tablas 4.4 y 4.6).

t_round	***t***	***x***	***y***	***z***	***Kitchen_AP***	***...***	***Study_AP***
46	46,012	1,082	-0,088	0,024	-75	...	NaN
46	46,061	1,072	-0,126	0,078	-68	...	NaN
46	46,111	1,186	-0,144	0,130	-68	...	NaN
46	46,161	1,044	-0,218	0,174	-68	...	NaN
46	46,212	0,976	-0,144	0,162	-68	...	NaN
46	46,261	0,940	-0,068	0,112	-67	...	NaN

Tabla 4.5: Ejemplo del archivo *acceleration.csv* de la secuencia 00001 de *train*

t_round	*x_mean*	*x_std*	...	*Kitchen_AP_mean*	...
46	0,9668	0,1152	...	-73,400000	...
47	0,3769	0,4145	...	-90,382854	...
48	0,6807	0,4851	...	NaN	...
49	0,8571	0,1126	...	-102,000000	...

Tabla 4.6: Resultados al aplicar groupby

Los archivos *video_hallway.csv*, *video_living_room.csv* y *video_kitchen.csv*, tienen una forma similar de distribución de los datos que *acceleration.csv*, por ello, se redondea por unidades cada valor de la columna *t* (tabla 4.7) a su marca temporal más cercana. Como consecuencia, se obtienen varios valores de información de coordenadas de los vídeos por cada marca temporal *(t_round)* y es necesario agrupar estos valores. La media y la desviación típica de cada grupo de cada atributo es el resultado del valor de las coordenadas para cada marca temporal (tabla 4.8).

t_round	*t*	*kn_centre_2d_x*	*kn_centre_2d_y*	...
23	22,983	273,498413	139,039902	...
23	23,014	276,435059	135,167404	...
23	23,077	279,988434	134,355499	...
23	23,108	280,044556	135,801361	...
23	23,139	280,899841	137,225235	...

Tabla 4.7: Ejemplo del archivo *video_kitchen.csv*
de la secuencia de datos 00001 de *train*

t_round	*kn_centre_2d_x_mean*	*kn_centre_2d_x_std*	...
21	0	0	...
22	244.472836	17,395877	...
23	285,605985	4,554205	...
24	283,438101	2,582146	...
25	283,545743	2,321926	...

Tabla 4.8: Resultados al aplicar groupby

El periodo de tiempo que el sujeto permanece en cada estancia de la casa inteligente, está registrado en el archivo *pir.csv*. Para unificar la ubicación del sujeto,

con los demás datos se realiza un *One Hot Encoding* sobre la variable *name* (tabla 4.10). Al aplicar esta técnica se elimina la variable nominal *name* y se agrega una variable binaria por cada categoría que tiene la variable nominal en cuestión [41]. En este caso, *name* tiene 9 categorías (tabla 4.9), por tanto se han añadido 9 variables binarias.

bath	*bed1*	*bed2*	*hall*	*kitchen*	*living*	*stairs*	*study*	*toilet*

Tabla 4.9: Categorías de la variable *name*

De este modo, si el sujeto está desde el instante 15,873 al instante 26,559 en el salón o *living* (teniendo en cuenta que dichos instantes se redondean por unidades a la marca temporal, creando las columnas *start_round* y *end_round*), se asigna el valor 1 a la columna *living* (tabla 4.11), creada al realizar *One Hot Encoding* sobre la columna *name* (tabla 4.10).

start	***start_round***	***end***	***end_round***	***name***	***index***
20,657	21	22,924	23	*stairs*	6
15,873	16	26,559	27	*study*	7
23,775	24	28,951	29	*kitchen*	4
30,457	30	33,776	34	*kitchen*	4

Tabla 4.10: Ejemplo del archivo pir.csv de la secuencia de datos 00001 de *train*

t_round	***bath***	***bed1***	***bed2***	***hall***	***kitchen***	***living***	***stairs***	***study***	***toilet***
21	0	0	0	0	0	0	1	1	0
22	0	0	0	0	0	0	1	1	0
23	0	0	0	0	0	0	0	1	0
24	0	0	0	0	1	0	0	1	0
25	0	0	0	0	1	0	0	1	0

Tabla 4.11: Después de hacer One-Hot Encoding

Una vez realizados estos cambios y transformaciones sobre los datos del archivo *pir.csv*, los atributos o columnas correspondientes a la tabla 4.11 se unen con el resto de datos descritos con anterioridad. Por otro lado, cabe recalcar que existe la posibilidad de superposición, debido a que el sujeto esté en el medio de dos estancias o que esté cerca del sensor de la estancia consecutiva, como ocurre en el periodo de tiempo que abarca desde la marca de tiempo 24 a la 27 (tabla 4.10). Y

se asigna el valor 1 a ambas estancias (tabla 4.11).

Por otro lado, se han añadido cinco tipos de variables: variables temporales, variables móviles, variables diferenciales, variables tendencia y variables probabilísticas en t-5 ($t = t\_round - 5$). Las variables temporales relacionan los valores del acelerómetro en los ejes (x, y, z) de cada marca de tiempo con cinco marcas antes (tablas 4.12 y 4.13).

x_mean	***x_std***	***y_mean***	***y_std***	***z_mean***	***z_std***
x_mean_t-1	*x_std_t-1*	*y_mean_t-1*	*y_std_t-1*	*z_mean_t-1*	*z_std_t-1*
x_mean_t-2	*x_std_t-2*	*y_mean_t-2*	*y_std_t-2*	*z_mean_t-2*	*z_std_t-2*
x_mean_t-3	*x_std_t-3*	*y_mean_t-3*	*y_std_t-3*	*z_mean_t-3*	*z_std_t-3*
x_mean_t-4	*x_std_t-4*	*y_mean_t-4*	*y_std_t-4*	*z_mean_t-4*	*z_std_t-4*
x_mean_t-5	*y_std_t-5*	*y_mean_t-5*	*y_std_t-5*	*z_mean_t-5*	*z_std_t-5*

Tabla 4.12: Variables temporales de información del acelerómetro

t_round	***x_mean***	***x_mean_t-1***	***x_mean_t-2***	***...***	***x_mean_t-5***
46	0,9668	0,9652	1,0001	...	0,9523
47	0,3769	0,9668	0,9652	...	0,9642
48	0,6807	0,3769	0,9668	...	0,9444
49	0,8571	0,6807	0,3769	...	1,0001
50	0,6775	0,8571	0,6807	...	0,9652

Tabla 4.13: Ejemplos de las variables temporales

Las variables móviles con ventana 6, se calculan haciendo la media de cada grupo de variables temporales (tabla 4.12) tal como queda reflejado en las fórmulas 4.1, para cada uno de los ejes (x, y, z).

$$xm\_mean\_v6 = \frac{x\_mean + x\_mean\_t{-}1 + x\_mean\_t{-}2 + ... + x\_mean\_t{-}5}{6}$$

$$xs\_mean\_v6 = \frac{x\_std + x\_std\_t{-}1 + x\_std\_t{-}2 + ... + x\_std\_t{-}5}{6} \tag{4.1}$$

...

x_ mean	*x_ mean_ t-1*	*x_ mean_ t-2*	*...*	*xm_ mean_ v6*
0,8875	0,6775	0,8571	...	0,74108
0,8668	0,6775	0,6775	...	0,72441
0,935	0,8668	0,8875	...	0,81743
0,9452	0,935	0,8668	...	0,86151
0,9436	0,9452	0,935	...	0,87593

Tabla 4.14: Ejemplos de las variables temporales

Las variables diferenciales, se computan con la diferencia entre distintas variables, tal como se indica en las fórmulas 4.2 y 4.3, para cada uno de los ejes (x, y, z).

$$\begin{aligned} xm\_dif\_t0-1 &= x\_mean - x\_mean\_t-1 \\ xm\_dif\_t0-5 &= x\_mean - x\_mean\_t-5 \\ xs\_dif\_t0v5 &= x\_mean - \frac{x\_mean\_t-1+...+x\_mean\_t-5}{5} \end{aligned} \tag{4.2}$$

$$\begin{aligned} xs\_dif\_t0-1 &= x\_std - x\_std\_t-1 \\ xs\_dif\_t0-5 &= x\_std - x\_std\_t-5 \\ xs\_dif\_v6 &= x\_std - \frac{x\_std\_t-1+...+x\_std\_t-5}{5} \end{aligned} \tag{4.3}$$

x_ mean	*x_ mean_ t-1*	*xm_ dif_ t0-1*	*xm_ trend01*
0,8875	0,6775	0,21	1
0,935	0,8668	0,0682	1
0,9452	0,935	0,0102	1
0,9436	0,9452	- 0,0016	0

Tabla 4.15: Ejemplo de la variable diferencial xm_dif_t0-1 y de la variable tendencia xm_trend01

Las variables tendencia complementan a las variables diferenciales, de tal manera que indican con un 1 si la diferencia es mayor que 0, con un -1 si es menor a cero y con 0 si al aproximarlo a la centésima la diferencia es 0 (tabla 4.15). Las variables probabilísticas en t-5, representan las probabilidades de cada actividad (tabla 4.1, página 26) cinco marcas de tiempo antes (tabla 4.16).

t_round	*a_ascend*	*a_ascend_t-5*	*a_descend*	*a_descend_t-5*
1043	0,55	0	0	0
1044	1	0	0	0
...	...	...	...	...
1048	0	0,55	0	0
1049	0,15	1	0	0

Tabla 4.16: Ejemplo de las variables probabilisticas en $t = -5$

Tras la integración, unificación, reducción y generación de nuevos atributos necesarios para un buen análisis predictivo, se forma una conjunto de datos de 228 columnas, con muchos *NaNs* o datos desconocidos, principalmente de las columnas que recogen la intensidad de señal recibida (RSSI, *Received Signal Strength Indicator*) por los puntos de acceso; y de las columnas que contienen el valor de las coordenadas que recogen las diferentes cámaras dispuestas por la casa 4.18).

Para reducir el posible ruido del análisis predictivo es necesario eliminar o reemplazar todos los *NaNs*, que representan mediciones erróneas o pérdidas, porque puede causar errores con algunos algoritmos de aprendizaje automático. La estrategia más sencilla es eliminar las filas que contienen *NaNs*, pero esto para columnas con muchos *NaNs* es demasiado restrictivo y provoca una gran pérdida de datos. Otra alternativa es reemplazarlos por otro valor. Algunas de la diferentes opciones para reemplazar los *NaNs* son:

- Un valor constante, que tenga un significado dentro del dominio.
- Un valor de otro registro seleccionado al azar.
- La media, mediana, moda de la columna.
- Un valor estimado por otros modelos predictivo.

Tras analizar el significado de los *NaNs* en las columnas de la tabla 4.18, se concluye que los *NaNs* de las columnas *AP* se van a sustituir por el valor constante -110. Un valor de RSSI por debajo de -80dBm indica que la señal recibida no es suficientemente buena [42], por tanto, se estima que -110 es una valor suficientemente bajo con el objetivo de que indique que para esos casos el punto de acceso no ha recibido una buena señal. Por otro lado, los *NaNs* de las columnas de los vídeos se van a sustituir por el valor constante 0, que indica que la cámara de vídeo no ha recogido ningún dato, porque el sujeto no se encuentra en esa habitación y por tanto, se considera que la cámara no está activada. Y el resto de *Nans* se elimina.

Original	Sin *NaNs*
16608	14680

Tabla 4.17: Información del tamaño de los datos

Columnas AP	Columnas de los videos
Kitchen_AP_mean, Kitchen_AP_std *Loungue_AP_mean, Loungue_AP_std* *Upstairs_AP_mean, Upstairs_AP_std* *Study_AP_mean, Study_AP_std*	*centre_2d_x_* / centre_2d_y_** *bb_2d_br_x_* / bb_2d_br_y_** *bb_2d_tl_x_* / bb_2d_tl_y_** *centre_3d_x_* / centre_3d_y_* / centre_3d_z_** *bb_3d_brb_x_* / bb_3d_brb_y_* / bb_3d_brb_z_** *bb_3d_flt_x_* / bb_3d_flt_y_* / bb_3d_flt_z_**

Tabla 4.18: Columnas AP y de los videos
*(* = hallway / kitchen / living)*

Los sensores ambientales de presencia pasivo por infrarrojos (PIR) utilizados durante el proceso anotación y recogida de datos, son de bajo coste y tienden a cometer fallos, por la presencia de radiación solar u otros aspectos ambientales, lo que provoca falsos positivos o negativos de presencia. Para disminuir la existencia de estos falsos positivos y teniendo en cuenta la ventaja de que en la casa solo se encuentra el sujeto, en las estancias con cámaras, se anulará la detección de sus correspondientes sensores si la cámara no recoge información. Y si la cámara está recogiendo información se anulará la detección del resto de sensores ubicado en las diferentes estancias de la casa.

Por último, se han creado dos variables (*targets_nominal_max* y *targets_max*) que representan, en formato nominal y numérico respectivamente, la actividad que tiene la mayor probabilidad según las columnas probabilísticas de la tabla 4.19 en la 38 (*a_ascend*, *a_descend*, *a_jump*..)

a_ascend	*a_descend*	*a_walk*	...	*targets_nominal_max*	*targets_max*
1	0	0	...	*a_ascend*	0
0,7	0	0,3	...	*a_ascend*	0
0,4	0	0,7	...	*a_walk*	4
0	0	1	...	*a_descend*	1

Tabla 4.19: Ejemplo de las variables *targets_nominal_max* y *targets_max*

Como se observa en la tabla 4.20 cada *targets_nominal_max* se corresponde con un *targets_max*.

targets_max	***targets_nominal_max***	***targets_max***	***targets_nominal_max***
0	*a_ascend*	10	*p_stand*
1	*a_descend*	11	*t_bend*
2	*a_jump*	12	*t_kneel_stand*
3	*a_loadwalk*	13	*t_lie_sit*
4	*a_walk*	14	*t_sit_lie*
5	*p_bent*	15	*t_sit_stand*
6	*p_kneel*	16	*t_stand_kneel*
7	*p_lie*	17	*t_stand_sit*
8	*p_sit*	18	*t_straighten*
9	*p_squat*	19	*t_turn*

Tabla 4.20: Correspondencia entre *targets_max* y *targets_nominal_max*

4.1.3. Síntesis de los atributos

Tras el pre-procesado completo de los datos se forma una conjunto de datos con 227 columnas, denominadas en los siguientes puntos de la siguiente manera:

- La columna de tiempo: representa la marca de tiempo correspondiente a cada actividad.

t_round

Tabla 4.21: Columna de tiempo

- Las columnas probabilísticas: son 20 y representan la probabilidad de reconocimiento de cada actividad. En la tabla 4.1, página 26 se especifican todas las actividades que se pueden reconocer.

a_ascend	***a_descend***	***a_jump***	***a_walk***	...

Tabla 4.22: Columnas probabilísticas

- Las columnas RSSI: son 8 y representan la media y la desviación típica de la intensidad de señal recibida (RSSI, *Received Signal Strength Indicator*) por los puntos de acceso de la cocina, del recibidor, de las escaleras y del estudio.

Kitchen_AP_mean	*Kitchen_AP_std*	*...*

Tabla 4.23: Columnas RSSI

- Las columnas probabilísticas en t-5 ($t = t\_round - 5$): son 20 y representan las probabilidades de cada actividad, cinco marcas de tiempo antes.

a_ascend_t-5	*a_descend_t-5*	*a_jump_t-5*	*...*

Tabla 4.24: Columnas probabilísticas en t-5

- Las columnas PIR: son 9 columnas y representan la información de activación y desactivación de los sensores ambientales de presencia pasiva por infrarrojo.

bath	*bed1*	*bed2*	*hall*	*kitchen*	*living*	*stairs*	*study*	*toilet*

Tabla 4.25: Columnas PIR

- Las columnas de los vídeos: forman un grupo de 90 columnas y representan, la media y la desviación típica de los valores de información coordenadas (x, y, z).

vh_bb_2d_br_x_mean	*vh_bb_2d_br_x_std*	*...*

Tabla 4.26: Columnas de los vídeos

- Las columnas temporales: son 36 columnas y representan diferentes valores de las coordenadas (x, y, z) de información de los puntos de acceso o receptores ubicados dentro de la casa.

x_mean	*x_std*	*x_mean_t-1*	*x_std_t-1*	*...*

Tabla 4.27: Columnas temporales

- Las columnas móviles: forman un grupo de 6 columnas y representan la media móvil con ventana 6 de las variables temporales (tabla 4.27).

xm_mean_v6	*xs_mean_v6*	*ym_mean_v6*	*ys_mean_v6*	*...*

Tabla 4.28: Columnas móviles

- Las columnas diferencias: forman un grupo de 18 columnas y representan la diferencia entre las variables temporales (tabla 4.27) y entre las variables móviles (tabla 4.28).

xm_dif_t0-1	*xm_dif_t0-5*	*xm_dif_t0v5*	*xs_dif_t0-1*	*...*

Tabla 4.29: Columnas diferencias

- Las columnas tendencia: forman un grupo de 18 columnas y representan diferentes valores de las coordenadas (x, y, z) de información de los puntos de acceso o receptores ubicados dentro de la casa.

xm_trend01	*xm_trend05*	*xm_trendv5*	*...*

Tabla 4.30: Columnas tendencias

- Las columnas targets: son 2 y representan la variable respuesta del conjunto de datos en formato nominal y numérico, respectivamente.

targets_nominal_max	*targets_max*

Tabla 4.31: Columnas targets

4.2. Análisis predictivo

En esta sección se detallan los modelos analizados en diferentes experimentos con el fin de determinar y optimizar los resultados; y de esta manera averiguar el movimiento que realiza el sujeto en cada marca de tiempo, utilizando los datos pre-procesados que se detallan en el apartado 4.1.2.

Los distintos experimentos se pueden dividir en tres fases. En la primera y segunda fase se realiza un análisis predictivo con la herramienta RapidMiner y Python, respectivamente. Y en la tercera fase se opta por reducir la dimensionalidad de los datos con Python.

4.2.1. Variable respuesta y criterios de evaluación

La variable respuesta u objetivo durante el entrenamiento de los diferentes algorítmos es *targets_nominal_max* y *targets_max* para RapidMiner y para Python, respectivamente. Por otro lado, para evaluar los diferentes modelos se ha utilizado la medida de evaluación *accuracy* y *Brier Score*, detallas en la sección 3.6 del capítulo 3.

En el calculo del *accuracy* se ha utilizado *targets_nominal_max* o *targets_max*, dependiendo de la herramienta; y se ha analizado la matriz de confusión que se ha obtenido con el modelo con el que se ha alcanzado el resultado más óptimo.

Para el cálculo del *Brier Score* se ha simulado una función en Python que calcula dicha medida de evaluación según los criterios de la competición *Senior Data Science: Safe Aging with SPHERE* [1] a la que pertenecen los datos tratados. Esto permite comparar los resultados de las pruebas con a otros participantes, ya que la competición está cerrada y no es posible publicar las predicciones obtenidas y que la competición las evalúe internamente.

```
def brier_score(target, predicted, class_weights):
return np.power(target - predicted, 2.0).dot(class_weights).mean()
```

Tabla 4.32: Función del *Brier Score* en Python

La frecuencia de datos para cada actividad de la variable *targets_nominal_max* está desequilibrada, como se puede observar en la figura 4.5. Por tanto, se ha fijado mayor peso (*class_weights*) en las clases menos frecuentes, según los criterios de la organización.

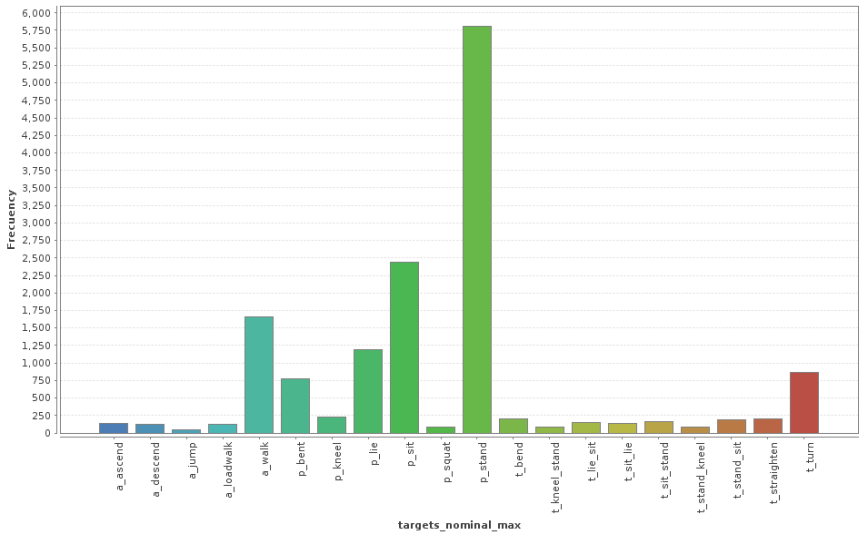

Figura 4.5: Frecuencia de las actividades

targets_ nominal_ max	*weights*	*targets_ nominal_ max*	*weights*
a_ ascend	1,35298455691	*p_ stand*	0,110181365961
a_ descend	1,38684574053	*t_ bend*	1,07803284435
a_ jump	1,59587388404	*t_ kneel_ stand*	1,36560417316
a_ loadwalk	1,35318713948	*t_ lie_ sit*	1,17024113802
a_ walk	0,347783666015	*t_ sit_ lie*	1,1933637414
p_ bent	0,661081706198	*t_ sit_ stand*	1,1803704493
p_ kneel	1,04723628621	*t_ stand_ kneel*	1,34414875433
p_ lie	0,398865222651	*t_ stand_ sit*	1,11683830693
p_ sit	0,207586320237	*t_ straighten*	1,08083910312
p_ squat	1,50578335208	*t_ turn*	0,503152249073

Tabla 4.33: *Weights* para cada clase

Tal como se pretendía, al comparar la figura 4.5 y la tabla 4.33 se observa que los pesos (*weights*) más altos corresponde con las actividades menos frecuentes, y viceversa, para compensar el desequilibrio entre las distintas actividades.

4.2.2. Análisis de los clasificadores con RapidMiner

En este apartado, se evalúan diferentes modelos de algunas familias de algoritmos como Bayes, Trees, Rule y Deep Learning con la herramienta RapidMiner, que permite modelar datos y construir modelos de forma sencilla y visual. Los detalles de dicha herramienta se desarrollan en la sección 3.7.

Las figuras 4.6, 4.7, 4.8 y 4.9 representan el proceso que se ha utilizado, para evaluar los diferentes modelos.

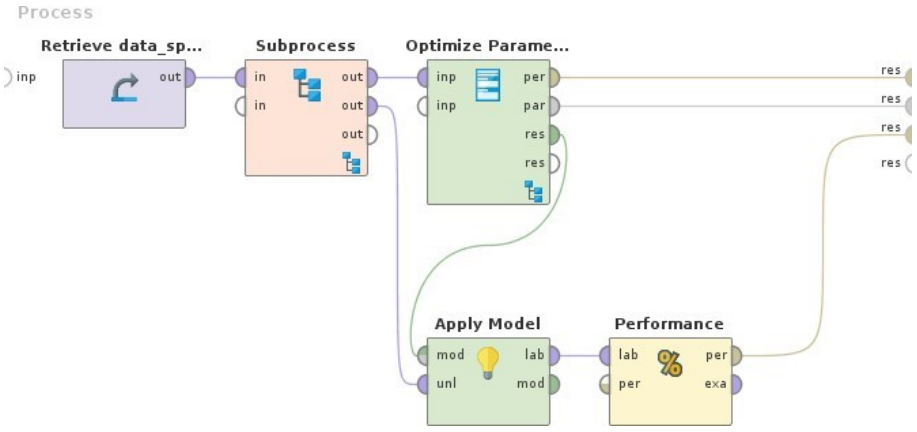

Figura 4.6: *Process* de RapidMiner

La figura 4.6, representa el procedimiento que se ha seguido para construir el modelo óptimo, de manera que tras cargar el conjunto de datos pre-procesado, tal como se detalla en el apartado 4.1.2, se ejecuta el módulo *Subprocess*, el *Optimize Parameters* y por último los operadores *Apply Model* y *Performance*.

El *Apply Model* se encarga de aplicar el modelo en el conjunto de datos de *test* y el *Performance* de evaluar el rendimiento del modelo sobre los datos *test* mediante la medida del *accuracy* (definida en la sección 3.6).

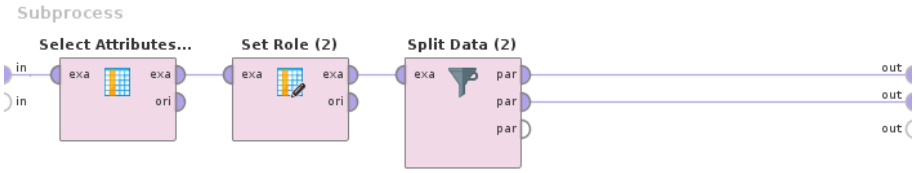

Figura 4.7: *Subprocess* de la figura 4.6 de RapidMiner

En la figura 4.7, se observan los diferentes *Operators* del módulo *Subprocess* de la figura 4.6. El operador denominado *Select Attributes*, selecciona el subconjunto de columnas o atributos siguientes:

- La columna *targets_ nominal_ max*.
- Las columnas probabilísticas en $t = -5$.
- Las columnas tendencia.
- Las columnas diferencias.
- Las columnas móviles.
- Las columnas temporales.
- Las columnas RSSI.
- Las columnas PIR.
- Las columnas de los vídeos.

Nota:

En la sección 4.1.3 hay una pequeña descripción de todas las columnas del conjunto de datos después del pre-procesado.

El operador *Set Role* de la figura 4.7, cambia el rol del atributo *targets_ nominal_ max* a *label*, para que se utilice como variable respuesta (sección 4.2.1). Por último, el operador *Split Data* permite dividir el conjunto de datos en dos subconjuntos o más, de acuerdo con los tamaños relativos que se especifiquen. Siguiendo el procedimiento la técnica *Holdout* (detallada en la sección 3.6) se divide el conjunto en dos subconjuntos (*test* y *train*) aleatoriamente, de manera que el primer conjunto representa el 80 % del total y el segundo el 20 % del total.

El operador *Optimize Parameters (Grid)* de la figura 4.6, se encarga de encontrar los valores óptimos de cualquier operador. En este caso, este operador se

encarga de ejecutar todas las combinaciones de valores para los parámetros, validarlas y deducir los valores óptimos. Para validar los parámetros del modelo se ha utilizado la validación cruzada, mediante el operador *Cross Validation*.

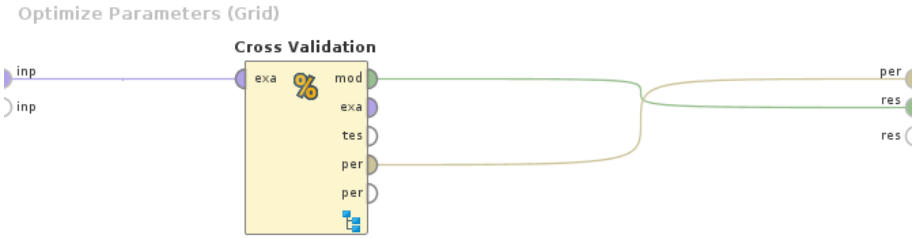

Figura 4.8: *Optimize Parameters* de la figura 4.6 de RapidMiner

El operador *Cross Validation* representado en la figura 4.8, divide los datos *train* en k subconjuntos de igual tamaño, de estos subconjuntos uno de ellos se retiene para utilizar como datos de validación y los otros como datos de entrenamiento. Este proceso se repite k veces utilizando cada subconjunto una vez como datos de validación. Los k resultados que se obtienen se promedian.

Este proceso se divide en dos partes, una de *training* y otra de *testing*, como se observa en la figura 4.9. En el primero se entrena el modelo con un algoritmo (por ejemplo con el algoritmo *Gradient Boosted Trees*) y en el segundo se evalúa dicho modelo con el subconjunto de datos de validación para cada iteración.

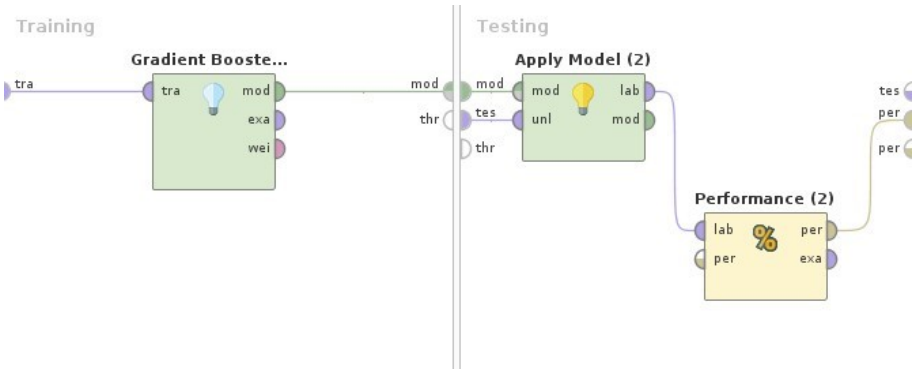

Figura 4.9: *Cross Validation* de la figura 4.6 de RapidMiner

El operador *Apply Model* de la figura 4.8 se encarga de aplicar el modelo en el conjunto de datos de *test* y el *Performance*, evalúa el rendimiento del modelo sobre los datos *test* mediante la medida del *accuracy* (definida en la sección 3.6).

Se ha aprovechado el mismo procedimiento para evaluar los distintos algoritmos cambiando el operador *Gradient Boosted Trees* por el operador correspondiente para cada uno de los siguientes algorítmos:

- *K-NN*
- *Decision Tree*
- *Random Forest*
- *Gradient Boosted Tree*
- *Rule Induction*
- *Deep Learning*

Nota:

Para determinar los valores óptimos de los parámetros de cada algoritmo se han realizado varias pruebas con diferentes rangos de valores para cada parámetro, como se observa en las tablas 4.34, 4.35, 4.36, 4.37, 4.38 y 4.39.

El *K-NN* es muy sensible a sus dos parámetros principales: k, que especifica el número de vecinos más cercanos; y *measure type*, que indica el tipo de métrica que se utiliza para calcular la cercanía de los k vecinos más cercanos, tras realizar diferentes experimentos con los valores de la tabla 4.34, se concluye que la distancia óptima es la *Manhattan* de tipo numérica, tal como está definida en la sección 3.5.

K-NN: 65,11 %	
k:	[1; 2; 3; 4; 5; 6; 7; 8; 9; 10; 25]
Measure types:	[*NumericalMeasure*; *MixedMeasure; BregmanDivergences*]
- Numerical measure:	[*EuclideanDistance; ChebychevDistance; ManhattanDistance*]
- Mixed measure:	[*MixedEuclideanDistance*]
- Divergence:	[*SquaredEuclideanDistance*]

Tabla 4.34: Valores especificados para *Grid* con *K-NN*
El valor óptimo de cada parámetro se marca de la siguiente manera: —

El *Decision Tree*, permite minimizar el riesgo y maximizar el *accuracy* porque calcula el valor esperado de cada opción del árbol. El parámetro *criterion* especifica el criterio que se utiliza para elegir los atributos y el orden en el que aparecen en el árbol, el tipo que se utiliza en todos los experimentos de este modelo, es el *information_gain*, porque permite divisiones de los nodos en múltiples caminos (sección 3.5). Otros parámetros son:

- *Maximal depth*, indica el límite de profundidad máxima que puede llegar a tener el árbol. Si se establece a -1 indica que no hay límite, por tanto, el árbol se construye hasta que se cumplan otros criterios de detención.
- *Pruning*, establece si el árbol va a ser podado, de acuerdo al nivel de confianza, o no. El parámetro que establece el nivel de confianza es denominado *confidence*.
- *Prepuning*, indica si se van a usar otros criterios para la detención, además de la profundidad (*maximal depth*), como:

- *Minimal gain*, indica la ganancia mínima que tiene que ganar un nodo para poder dividirse.
- *Minimal leaf size*, establece el número mínimo de ejemplos que debe tener una hoja del árbol.
- *Minimal size for split*, establece el número mínimo de ejemplos que debe tener un nodo para dividirse.
- *Minimal of prepruning alternatives*, ajusta la cantidad de nodos alternativos que se probarán si la división de un cierto nodo no favorece.

Decision Tree: 65,04 %	
Criterion	[*information_gain*]
Maximal depth of tree:	[*-1*; *50*]
Apply_pruning: *- Confidence:*	[*true*; *false*] [*1E-7*; *1E-5;1E-3*]
Apply_prepruning: *- Minimal gain:* *- Minimal leaf size:* *- Minimal size for split:* *- Number of prepruning alternatives:*	[*true*; *false*] [*0*; *8; 16; 24; 32; 40*] [*1; 2; 4; 7;* *8*; *9; 10; 12; 19; 40; 51*] [*10; 37* *40*; *42; 45; 50; 70; 100*] [*0*; *10; 20; 30; 40; 50*]

Tabla 4.35: Valores especificados para *Grid* con *Decision Tree*
El valor óptimo de cada parámetro se marca de la siguiente manera: —

El *Random Forest*, es un tipo de *Ensemble Tree* que combina diferentes árboles con inicializaciones aleatorias e independientes. El número de árboles se especifica con el parámetros *number of trees*, para cada árbol se selecciona un subconjunto de ejemplos. Si se selecciona a *true* el parámetros *enable parallel execution*, los subconjuntos se entrenan en paralelo, a través de subprocesos de la CPU disponibles (esta ejecución simultánea puede provocar problemas de memoria).

Los parámetros *criterion*, *maximal depth*, *apply prepruning*, *minimal gain*, *minimal leaf size*, *minimal size for split*, *number of prepruning alternatives*, *apply pruning*, *confidence*, tienen la misma función que en el modelo *Decision Trees*. Algunos otros son:

- *Guess subset ratio*, si se fija como *false*, se utiliza el valor del *subset ratio* para elegir el número de atributos aleatoriamente.

  especificar la porción con el parámetro *subset ratio* y utiliza dicha porción para elegir el número de atributos aleatorios
- *Voting strategy*, especifica la estrategia de predicción. Estas puede ser:

  - *Confidence vote*, selecciona la clase que tiene la mayor confianza acumulada.

  - *Majority vote*, selecciona la clase que fue predicha por la mayoría de los modelos de árbol.

- *Use local random seed*, indica si se usa una semilla local aleatoria para la aleatorización. El valor de dicha semilla se especifica en el parámetro *local random seed*.

Random Forest: 70,53 %	
Number of trees	[*20;* *100*]
Criterion	[*information_gain*]
Maximal depth of tree:	[*5; 40;* *100*]
Apply_pruning: *- Confidence:*	[*true* *; false*] [*1.0E-7*]
Apply_prepruning: *- Minimal gain:* *- Minimal leaf size:* *- Minimal size for split:* *- Number of prepruning alt*ernatives:	[*true* *; false*] [*0* *; 15*] [*0* *; 1; 10*] [*0* *; 1; 10*] [*15*]
Guest subset ratio: *- Subset ratio:*	[true; *false*] [*0,8*]
- Voting Strategy:	[*confidence vote* *; majority vote*]
Use local random seed: *- Local random seed:*	[true; *false*] [*53*]

Tabla 4.36: Valores especificados para *Grid* con *Random Forest*
El valor óptimo de cada parámetro se marca de la siguiente manera: *—*

El *Gradient Boosted Trees*, es otro tipo de *Ensamble Tree* que combina diferentes árboles que obtienen resultados a través de estimaciones mejoradas gradualmente. Los parámetros *number of trees*, *maximal depth* y *use local random seed*, *local random seed*, tienen la misma función que en los modelos *Decision Trees* y *Random Forest*. Algunos otros son:

- *Reproducible*, si se establece a *true* especifica que el modelo debe ser reproducible, de forma aleatoria o no dependiendo del valor del parámetro *use local random seed*.
- *Maximum number of threads*, controla el nivel de paralelismo en la construcción del modelo, si se establece a *true*.
- *Min rows*, especifica el mínimo número de filas para que un nodo sea una hoja.
- *Min split improvement*, permite reducir el error cuadrático para que ocurra una división.

- *Number of bins*, especifica el número mínimo de contenedores del histograma que se va a construir durante el entrenamiento.
- *Learning rate*, permiten mejorar los modelos, con tasas pequeñas pero aumentan el tiempo de entrenamiento porque requiere más iteraciones.
- *Sample rate*, especifica la tasa de muestreo de fila del árbol.
- *Distribution*, indica la función de distribución para los datos de entrenamiento. Si se selecciona *AUTO*, se utiliza una distribución multinominal para etiquetas nominales y gausiana para numéricas.
- *Early stopping*, selecciona si se va a permitir la detención anticipada, en función de los siguientes parámetros: *stopping metric* y *stopping tolerance*.
- *Max runtime seconds*, especifica el tiempo máximo de ejecución permitido en segundos para la formación del modelo. Si se establece a 0 esta función queda deshabilitada.

  Existen otros parámetros, denominados *expert parameters*, para un ajuste más fino del algoritmo que no han sido analizados.

Gradient Boosted Trees: 74,68 %	
Number of trees:	[*300*; *600*; *900*]
Reproducible:	[*false*]
Maximal depth:	[*5*; *10*; *100*]
Min rows:	[*10*]
Min split improvement:	[*0*; *1*]
Number of bins:	[*20*; *100*]
Learning rate:	[*0,1*; *1*]
Sample rate:	[*0.1*; *1*]
Distribution:	[*AUTO*]
Early stopping:	[*false*]
Max runtime seconds:	[*0*]

Tabla 4.37: Valores especificados para *Grid* con *Gradient Boosted Trees*
El valor óptimo de cada parámetro se marca de la siguiente manera: —

Rule Induction, es un algoritmo basado en reglas (seccion 3.5) que comienza a analizar y a crear reglas con las clases menos frecuentes y de forma iterativa las reglas van creciendo y cortándose, hasta que todos los ejemplos se analizan. Los parámetros *criterion*, *sample ratio* y *use local random seed*, *local random seed*, tienen la misma función que en los modelos *Decision Trees*, *Random Forest* y *Gradient Boosted Trees*. Algunos otros son:

- *Pureness*, este parámetro especifica la pureza deseada, es decir, la relación mínima de la clase principal en un subconjunto cubierto para considerar el subconjunto puro.
- *Minimal prune benefit*, este parámetro especifica el valor mínimo de beneficio que tiene que tener el modelo para permitir la poda.

Rule Induction: 62,66 %	
Criterion	[*information_gain*]
Sample ratio	[*0,547; 0,666; 0,774; 0,8887; 1*]
Pureness:	[*0,8; 1*]
Minimal prune benefit:	[*0,8; 1*]
Use local random seed: *- Local random seed:*	[*false; true*] [*1; 26; 51; 75; 100*]

Tabla 4.38: Valores especificados para *Grid* con *Rule Induction*
El valor óptimo de cada parámetro se marca de la siguiente manera: —

El algoritmo *Deep Learning* se basa en una red neuronal artificial *feed-forward* de múltiples capas que está entrenada con un descenso de gradiente estocástico mediante retro-propagación. Los parámetros *reproducible*, *use local random seed*, *local random seed*, *early stopping*, *max runtime seconds expert parameters* tienen la misma función que en los modelos *Decision Trees* y *Random Forest*. Algunos otros son:

- *Activation*, especifica la función de activación, no lineal, que se utiliza para las capas ocultas. La más común es la *Rectifier*, que elige el valor máximo de (0,x) donde x es el valor de entrada.
- *Hidden layer sizes*, es el número y el tamaño de cada capa oculta.
- *Epochs*, especifica cuántas veces debe iterarse el conjunto de datos.
- *Compute variable importances*, si se establece a *True* indica que se debe calcular las *importances* variables para las características de entrada.
- *Train samples per iteration*, establece el número de datos *train* que se procesarán por iteración.
- *Adaptative rate*, el aprendizaje adaptativo permite combinar automáticamente los beneficios de la velocidad de aprendizaje y el entrenamiento de momento para evitar la convergencia lenta. Si se activa *True* es necesario especificar dos parámetros (*Epsilon* y *Rho*), que simplifican la búsqueda del valor óptimos de hiper parámetros.
  - *Epsilon*, es similar a *learning rate* durante el entrenamiento inicial y al momento en etapas posteriores. Su valor típico está entre 1E-10 y 1E-4.
  - *Rho*, es similar al momento y su valor típico está entre 0,9 y 0,999.

- *Standardize*, si está a *True* los datos se estandarizan automáticamente.
- *L1*, representa un método de regularización que restringe el valor absoluto de los pesos y tiene el efecto neto de eliminar algunas ponderaciones de un modelo para reducir la complejidad y evitar el sobreajuste.
- *L2*, es un método de regularización que restringe la suma de los pesos al cuadrado. Este parámetro introduce cierto sesgo en las estimaciones, pero a menudo produce ganancias sustanciales en el modelo.
- *Max w2*, representa el valor máximo de los pesos entrante al cuadrado en cualquier neurona. Si se establece a 0 significa que el valor máximo es infinito.
- *Loss function*, representa la función de pérdida (error) que minimiza el modelo.
- *Distribution function*, establece la función de distribución para los datos de entrenamiento. Si se fija a $AUTO$ la selección es automática.
- *Missing value handling*, establece el manejo que se va a hacer con los NaNs o valores perdidos. Se diferencian dos valores:
  - *Skip*: los NaNs se omiten.
  - *Mean imputation*, los NaNs se reemplazan por el valor medio.

Deep Learning: 71,48 %	
Activation:	[*Rectifier*]
Hidden layer sizes:	[*[120; 70; 40; 20]*]
Reproducible:	[*false*]
Epochs:	[*10; 73,33; 136,667;* *200* *; 400; 600*]
Compute variable importances:	[*false*]
Train samples per iteration:	[*-2*]
Adaptive Rate: *- Epsilon:* *- Rho:*	[*true*] [*1E-10* *; 1E-7; 1E-8; 1E-3*] [*0,99;* *0,99*]
Standardize:	[*true* *; false*]
L1:	[*1E-4;* *1E-3* *; 1E-1*]
L2:	[*0;* *5E-3* *; 1E-2*]
Max w2:	[*10*]
Loss function:	[*Automatic*]
Distribution function:	[*AUTO*]
Misssing values handling:	[*Meanimputation*]
Max runtime seconds:	[*0*]

Tabla 4.39: Valores especificados para *Grid* con *Deep Learning*
El valor óptimo de cada parámetro se marca de la siguiente manera: —

Tras evaluar cada uno de estos algorítmos, se concluye que el algoritmo que mejor se adapta a este conjunto de datos es el *Gradient Boosted Trees*, tal como se observa en la tabla 4.40, que se muestran los resultados obtenidos con cada uno de los algoritmos analizados.

Familia	**Clasificador**	**Accuracy**
Lazy	*K-NN*	65,11 %
Trees	*Decision Tree*	65,04 %
Trees	*Random Forest*	70,53 %
Trees	*Gradient Boosted Trees*	74,68 %
Rule	*Rule Induction*	62,88 %
ANN	*Deep Learning*	71,48 %

Tabla 4.40: Resultados de cada algoritmo con RapidMiner

La matriz de confusión del *Gradient Boosted Trees* representada en la tabla

4.41, contiene información sobre las clasificaciones etiquetadas y las predichas para cada una de las clases. Los números a lo largo de la diagonal superior izquierda a la inferior derecha de la matriz de confusión representada en la tabla 4.41 equivalen al número de instancias clasificadas correctamente de cada clase.

	true p_stand	true a_walk	true t_turn	...	a_loadwalk	Precisión (%)
pred. p_stand	1064	125	97	...	3	74,00 %
pred. a_walk	50	180	34	...	8	56,07 %
pred. t_turn	13	5	18	...	0	39,13 %
pred. t_bend	2	0	0	...	0	53,33 %
pred. p_bent	4	1	2	...	0	76,06 %
pred. t_straight	1	1	0	...	1	54,55 %
pred. t_stand_k	0	0	1	...	0	60,00 %
pred. p_kneel	2	0	0	...	0	76,47 %
pred. t_kneel_s	2	0	0	...	0	54,55 %
pred. t_stand_s	6	0	0	...	0	61,90 %
pred. p_sit	3	4	3	...	0	90,67 %
pred. t_sit_lie	0	0	0	...	0	70,00 %
pred. p_lie	0	0	12	...	0	88,09 %
pred. t_lie_sit	0	0	4	...	0	50,00 %
pred. t_sit_sta	1	1	2	...	0	62,96 %
pred. a_descend	1	4	0	...	0	50,00 %
pred. a_ascend	0	7	0	...	0	65,22 %
pred. p_squat	1	0	0	...	0	80,00 %
pred. a_jump	0	1	0	...	0	80,00 %
pred. a_loadwalk	0	3	0	...	14	73,68 %
Recall (%)	92,60 %	54,22 %	10,40 %	...	53,85 %	

Tabla 4.41: Matriz de confusión del *Gradient Boosted Trees*

4.2.3. Análisis de los clasificadores con Python

El algoritmo con el que se obtiene el mejor resultado en el apartado anterior es el *Gradient Boosted Trees* (GBT), tal como se observa en la tabla 4.40. Por tanto, en este apartado se va a evaluar dicho algoritmo según el criterio *Brier Score* definido en la competición en código Python desarrollado en el apartado 3.6.

Para construir un modelo con el algoritmo GBT con Python se utiliza la biblioteca *scikit-learn* (sección 3.7), la cual está diseñada para trabajar con la biblioteca númerica *NumPy*, además para leer los datos se ha utilizado la biblioteca *Pandas*. Tras leerlos, se distribuye en tres DataFrame[1]: X , Y e y_ind; en Y se incluyen 20 columnas que representan la probabilidad de cada actividad; y_ind representa la variable respuesta o *targets* que se utiliza para entrenar el modelo; el resto de datos se incluyen en X. Para dividir los DataFrame en *train* y *test*, es decir,

[1]DataFrame: es una lista de vectores de igual longitud que se usa para almacenar tablas de datos

en datos de entrenamiento y de prueba, de forma aleatoria se utiliza la función *train_test_split*. La proporción de datos de *test* queda especificada por el parámetro *test_size* que se establece a 0,2, lo que significa que el 20 % de los datos serán categorizados como datos *test* y el 80 % como de entrenamiento (funciona de forma similar al operador *Split Data* en RapidMiner).

La función *GridSearchCV* se encarga de hacer una búsqueda exhaustiva de los valores de los parámetros especificados del algoritmo GBT de forma similar al operador *Optimize Parameters (Grid)* en RapidMiner. Para la evaluación del modelo se ha utilizado la técnica de *Cross-validation*, en vez de *Holdout* que es el método utilizado en los experimentos con RapidMiner. Por otro lado, la función *GradientBoostingClassifier* construye el modelo de forma similar al operador *Gradient Boosted Trees* de RapidMiner. En concreto, los parámetros de la tabla 4.42 tienen significados iguales o similares [43].

Python	**RapidMiner**
N estimators	*Number of trees*
Max depth	*Maximal depth*
Min samples split	*Min rows*
Learning rate	*Learning rate*
Subsample	*Sample rate*

Tabla 4.42: Correspondencia entre los parámetros de GBT en Python y en RapidMiner

A continuación, se detallan los parámetros del *Gradient Boosted Trees* en Python:

- *N estimators*, establece el número de etapas o árboles. Como este algoritmo es muy robusto a un ajuste excesivo de este parámetro, generalmente un valor alto da como resultado un mejor rendimiento.
- *Max depth*, establece la profundiad máxima del árbol.
- *Min samples split*, fija el número mínimo de muestras que debe haber para dividir un nodo interno.
- *Learning rate*, especifica la tasa de aprendizaje que reduce la contribución de cada árbol.
- *Subsample*, especifica el número de muestras o filas que utilizará cada árbol.

Gradient Booted Trees: 73 %	
N estimators:	[*300*]
Maximal depth:	[*5*; *10*]
Min samples split:	[*0,1*; *2*; *4*]
Learning rate:	[*0,1*; *1*]
SubSample:	[*0,8*; *1*]

Tabla 4.43: Valores de los parámetros del *Gradient Boosted Trees* en Pyhton

El *Brier Score* es una medida de evaluación que permite medir la precisión de las predicciones probabilísticas, es decir, la precisión de las probabilidades predichas de las actividades al compararlas con las 20 columnas probabilísticas definidas con anterioridad en la tabla 4.22 (de la página 38). Para calcular las probabilidades de las actividades predichas se utiliza la función *predict_proba* de la biblioteca *sklearn* de Python. Los resultados de las métricas de evaluación se muestran en la tabla 4.44.

Accuracy	***Brier Score***
73 %	16,6 %

Tabla 4.44: Resultados del GBT en Python

En conclusión, en *accuracy* es similar al obtenido con la herramienta RapidMiner con el algoritmo *Gradient Boosted Trees* (GBT) y el *Brier Score* se considera que es suficientemente bueno porque al compararlo con el *ranking* de la competición *Senior Data Science: Safe Aging with SPHERE* se ha alcanzado la posicón 20 de 579. Dicha posición está por encima de los 4 % mejores resultados.

4.3. Reducción de la dimensionalidad de los datos

Reducir la dimensionalidad de los datos puede ayudar a mejorar los resultados, aumentar el rendimiento y reducir la cantidad de ruido. En un conjunto de datos, hay muchas formas de reducir la cantidad de datos pero la mayoría de estas técnicas se incluyen en una de estos dos tipos: eliminación de características y extracción de características. A lo largo de este capítulo se detallan las diferencias entre ambas.

4.3.1. Eliminación de características

La eliminación de características consiste en involucrar en el entrenamiento solo las variables que son útiles y excluir las redundantes empleando técnicas que examinan la relación entre las características y la respuesta objetivo o etiqueta (sección 4.2.1). Algunas de las ventajas de estas técnicas son la simplicidad y la conservación original de las características. Sin embargo, hay que tener en cuenta, que al eliminar una variable se pierde toda la información y beneficios que aportaba el modelo [44].

La biblioteca *scikit-learn* proporciona funciones como *SelectKBest* que se encargan de eliminar las características según la función de puntuación y el número de características a seleccionar que se indique como parámetros de entrada (*score_func* y *k*). En clasificación, algunas de las funciones de puntuación que se pueden utilizar para el parámetro de entrada *score_func* son: *f_classif* o *mutual_info_classif*.

La función de puntuación *f_classif* se basa en el método *F-test*, que consiste en estimar el grado de dependencia lineal entre dos variables. En cambio los métodos de información mutua, como el de la función *mutual_info_classif* son capaces de capturar cualquier tipo de dependencia estadística, pero dado que no son paramétricos, necesitan más muestras para obtener una estimación precisa, en comparación con la primera función.

SelectKBest(score_func=<function f_classif>, k=10

Tabla 4.45: Función *SelectKBest* con los valores por defecto

4.3.2. Extracción de características

En la extracción de características, se crean grupos de variables independientes que son combinaciones de todas las variables originales, de manera, que se conservan las partes más valiosas de las antiguas variables. El análisis de componentes principales (PCA) es una técnica para la extracción de características, que permite identificar patrones en los datos y transformarlos de tal manera que se resalten sus similitudes y diferencias. Una vez que se han encontrado los patrones, las variables se combinan descartando las variables "menos importantes" al mismo tiempo que conservamos las partes más valiosas de todas las variables, porque las nuevas variables se crean intentando que expliquen la mayor cantidad de varianza de las originales, de esta manera se reduce el número de dimensiones sin mucha pérdida de información. [44].

La varianza explicada, se define como la relación entre la varianza de cada componente y la varianza total de todos los componentes principales. El número de componentes principales es variable y se puede ajustar dependiendo de los resultados que se quieran obtener, teniendo en cuenta que la varianza estimada y

que el número de componentes principales tiene que ser igual o menor al número de características originales.

Para aplicar esta técnica sobre los datos la biblioteca *scikit-learn* proporciona la función PCA, definida de la siguiente manera.

PCA(n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0,0, iterated_power='auto', random_state=None)

Tabla 4.46: Función PCA con los valores por defecto

4.3.3. Aplicación de técnicas de reducción de la dimensionalidad

Con el objetivo de mejorar los resultados y el rendimiento del modelo obtenido utilizando el algoritmo *Gradient Boosted Trees* sobre los datos de la competición *Senior Data Science: Safe Aging with SPHERE* se han realizado varios experimentos utilizando las técnicas (eliminación de características y extracción de funciones) de reducción de la dimensionalidad detalladas en los apartados anteriores. Para ello se han utilizando las funciones *PCA()* y *SelectKBest()*, además de *GridSearchCV()* de la biblioteca *scikit-learn* para realizar la búsqueda exhaustiva y comparativa del número de atributos (k) o componentes ($n\_components$) con los que se obtiene el resultado óptimo mediante validación cruzada.

Por otro lado, se ha analizado la varianza de las componentes principales al aplicar un PCA con distintos números de componentes principales, concluyendo que al reducir la dimensión más de un 85 %, de manera que el número de características se compriman de 211 a 30, se obtiene una varianza total mayor a $0,99988$. Lo que significa que con este número de componentes aún se conserva el $99,99988\,\%$ de la varianza de la variables originales, lo que indica una pérdida mínima de información.

Eliminación de características			**Extracción de funciones**	
k	**SelectKBest *f_classif***	**SelectKBest *mutual_info_classif***	***n_components***	**PCA**
30	62,96 %	68,32 %	30	60,80 %
100	71,32 %	70,96 %	100	68,54 %
150	72,01 %	71,85 %	150	70,01 %
190	72,93 %	73,11 %	190	69,86 %

Tabla 4.47: Resultados de accuracy al reducir la dimensionalidad con las diferentes técnicas

En la tabla 4.47 se observa que los resultados de *accuracy* obtenidos para un

PCA de 30 componentes son bastante bajos con respecto al resultado de *accuracy* de 73 % en el apartado anterior; a pesar de que la varianza conserve el 99, 99988 % de la varianza de la variables originales. El resto de experimentos no aporta ningún *accuracy* mayor a 73 % al anterior, por tanto, se concluye que ninguna de las técnicas tratadas son adecuadas para mejorar los resultados en este conjunto de datos.

5 | Escenario 2

Para complementar el análisis predictivo de los conjuntos de datos *Senior Data Science: Safe Aging with SPHERE* [1] se ha seleccionado otro conjunto de datos del repositorio de aprendizaje automático UCI, denominado *Smartphone-Based Recognition of Human Activities and Postural Transitions Data Set* [19].

Siguiendo la metodología CRISP-DM (al igual que con el primer conjunto de datos), el objetivo final que se fija (fase *Business Understanding*) es evaluar el reconocimiento de los movimientos utilizando la información de los sensores incorporados en los Smartphone, como los sensores GPS, de visión (como cámaras), de audio (como micrófonos), de luz, de temperatura, de dirección (como brújulas) o de aceleración (como acelerómetros); y de esta manera ahorrar costes.

5.1. Estudio de los datos

En esta sección se desarrolla los detalles y procedimiento que se ha seguido para el tratamiento de este conjunto de datos, lo que corresponde a la fases *Data Understanding* y *Data Preparation* de la metodología CRISP-DM.

Este conjunto de datos, es una versión actualizada de *Human Activity Recognition Using Smartphones Data Set* [20], que utiliza las señales originales de los sensores de los smartphone, en vez de las señales preprocesadas por una aplicación de filtro de ruido que se usaba en la primera versión, con el fin de hacer pruebas en línea con los datos en bruto.

5.1.1. Descripción del conjunto datos

En el proceso de extracción de este conjunto de datos, participaron 30 voluntarios de entre 19 a 48 años, los cuáles realizaron un serie de ejercicios compuestos por seis actividades básicas, tres estáticas (de pie, sentado y tumbado), tres dinámicas (caminar, bajar escaleras y subir escaleras) y otras seis transitorias que ocurrieron entre las actividades estáticas [20]. En la tabla 5.1 puede observar la etiqueta nominal y numérica a la que le corresponde cada actividad.

Para la recolección de datos fue necesario que todos los participantes llevasen un smartphone de alta gama (*Samsung Galaxy SII*) en la cintura durante la eje-

Tipo	**Actividad**	**Etiqueta**	
		Nominal	**Numérica**
Dinámica	Caminar	*WALKING*	1
	Subir escaleras	*WALKING_UPSTAIRS*	2
	Bajar escaleras	*WALKING_DOWNSTAIRS*	3
Estática	Sentado	*SITTING*	4
	De pie	*STANDING*	5
	Tumbado	*LAYING*	6
Transitoria	De pie a sentado	*STAND_TO_SIT*	7
	De sentado a de pie	*SIT_TO_STAND*	8
	De sentado a tumbado	*SIT_TO_LIE*	9
	De tumbado a sentado	*LIE_TO_SIT*	10
	De pie a tumbado	*STAND_TO_SIT*	11
	De tumbado a de pie	*LIE_TO_STAND*	12

Tabla 5.1: Descripción de las 12 actividades

cución del experimento, que capturó la aceleración lineal de 3 ejes y la velocidad angular de 3 ejes a una velocidad constante de $50Hz$ utilizando el acelerómetro y el giroscopio integrados en este móvil. Los experimentos fueron grabados en vídeo y etiquetados con las actividades correspondientes manualmente.

Las señales del acelerómetro y del giroscopio (*tAcc-XYZ* y *tGyro-XYZ*) en bruto fueron filtradas con un filtro mediana y un filtro Butterworth de paso bajo de tercer orden con frecuencia de corte (f_c) a $20Hz$ para eliminar el ruido. De forma similar la señal de aceleración se separó en señales de aceleración de cuerpo y gravedad utilizando otro filtro de paso bajo con una frecuencia de corte (f_c) a $0,3Hz$ (*tBodyAcc-XYZ* y *tGravityAcc-XYZ*) [20].

Después se calculó la aceleración lineal y la velocidad angular del cuerpo, de manera que se obtuvieron las señales Jerk[1]. Además de la magnitud de estas señales tridimensionales utilizando la distancia Euclídea (*tBodyAccMag*, *tGravityAccMag*, *tBodyAccJerkMag*, *tBodyGyroMag* y *tBodyGyroJerkMag*). Finalmente se aplicó una transformada rápida de Fourier (Fast Fourier Transform, *FFT*) a algunas de estas variables (*fBodyAcc-XYZ*, *fBodyAccJerk-XYZ*, *fBodyGyro-XYZ*, *fBodyAccJerkMag*, *fBodyGyroMag* y *fBodyGyroJerkMag*; el prefijo *f*, indica que son señales en el dominio de la frecuencia) [20].

[1]Jerk: es la tasa de cambio de aceleración; es decir, la derivada de la aceleración con respecto al tiempo

Señales	
Dominio del tiempo	**Dominio de la frecuencia**
tBodyAcc-XYZ	*fBodyAcc-XYZ*
tGravityAcc-XYZ	
tBodyAccJerk-XYZ	*fBodyAccJerk-XYZ*
tBodyGyro-XYZ	*fBodyGyro-XYZ*
tBodyGyroJerk-XYZ	
tBodyAccMag	*fBodyAccMag*
tGravityAccMag	
tBodyAccJerkMag	*fBodyAccJerkMag*
tBodyGyroMag	*fBodyGyroMag*
tBodyGyroJerkMag	*fBodyGyroJerkMag*

Tabla 5.2: Lista de señales que se usaron para estimar las características

Las distintas señales de la tabla 5.2 se utilizaron para estimar las distintas características del conjunto de datos como la media, la desviación típica, la desviación de la media absoluta, el máximo, el mínimo y otras; mediante las siguientes funciones [20]:

- mean()
- std()
- mad()
- max()
- min()
- sma()
- energy()
- iqr()
- entropy()
- arCoeff()
- correlation()
- maxInds()
- meanFreq()
- skewness()
- kurtosis()
- bandsEnergy()
- angle()

Después de preprocesar las señales del sensor (acelerómetro y giroscopio) mediante la aplicación de filtros de ruido, se tomaron muestras en ventanas móviles de ancho fijo de 2,56 segundos y 50 % de superposición (128 lecturas/ventana). Desde cada ventana, se obtuvo un vector de 561 características con variables del dominio de tiempo y frecuencia [19].

El conjunto de datos está dividido en dos partes que se pueden usar por separado. Por un lado, se proporciona los datos de los sensores en bruto diferenciados para cada participante y por otro lado, se proporcionan los registros de todas actividades realizadas por los sujetos. A su vez esta última parte del conjunto, está dividida en dos conjuntos: datos de entrenamiento (*train*) y datos de prueba (*test*), y por tanto será la uitilizada para el análisis predictivo [20]. Cada registro está compuesto por la siguiente información distribuida en diferentes archivos de

formato txt, como se representa en la tabla 5.3.

Archivo		Información
Train	***Test***	
X_train.txt	*X_test.txt*	Un vector con 561 atributos.
y_train.txt	*y_test.txt*	La etiqueta de la actividad asociada.
subject_id_train.txt	*subject_id_test.txt*	Un identificador del sujeto que realizó el experimento.

Tabla 5.3: Archivos del segundo conjunto de datos

5.2. Análisis predictivo

En esta sección se detallan los modelos analizados con este conjunto de datos (fase *Modeling* de CRISP-DM); con el objetivo de determinar si es viable utilizar los sensores de los propios Smartphone para complementar el análisis anterior.

5.2.1. Variable respuesta y criterios de evaluación

La variable respuesta de este conjunto de datos, tanto de la parte de datos *train* como de los de *test* se incluye en archivo *y_train.txt* e *y_test.txt*, respectivamente.

Por otro lado, para evaluar qué modelo se adapta mejor al objetivo de reconocer el movimiento se ha utilizado el *accuracy* que es una medida de evaluación que se ha detallado con anterioridad en la sección 3.6 de este Trabajo Fin de Grado.

5.2.2. Análisis de los clasificadores con RapidMiner

Para el análisis predictivo se ha elaborado un diseño final más simple, incluyendo en él, todos los modelos con los que se obtienen los mejores resultados. Dado que se han obtenido muy buenos resultados no se ha aplicado ninguna de la técnicas de optimización de los parámetros, ni de reducción de características utilizadas con la conjunto de datos analizada en el capítulo 4.

Las figuras 5.1 y 5.2 representan el proceso que se ha utilizado para evaluar los siguiente algoritmos detallados en la sección 3.5 de la página 14.

- *K-NN*
- *Naive Bayes*
- *Gradient Boosted Trees*
- *Deep Learning*
- *W - Simple Logistics*
- *W - REP Tree*

Antes de construir los modelos con los operadores que corresponden a cada algoritmo, ha sido necesario unificar los diferentes archivos descritos en la tabla 5.3 y etiquetar los registros. Para ello, se han definido dos *Subprocess* uno para los datos *train* y otro para los *test*, ambos con la misma estructura.

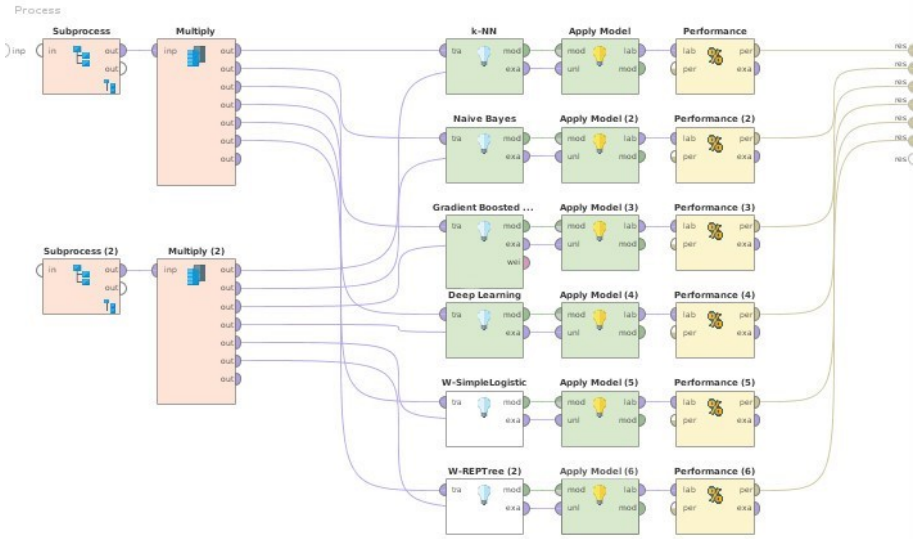

Figura 5.1: *Process* de RapidMiner

Tras cargar los tres archivos en los subprocesos, se ha generado un ID para cada uno de los archivos con el operador *Generate ID*, de manera que, al unirlos sus filas se relacionen de la forma correcta. Para unir los datos se ha utilizando el operador *Join*, en el orden en el que se observa en la figura 5.2. Las etiquetas que tiene el archivo *y_test* están en formato numérico, para que se pueda usar el mismo operador de evaluación (*Performance*) utilizado en el análisis del apartado 4.2.2, es necesario que las etiquetas sean nominales. Por tanto, se ha definido para cada valor numérico su correspondiente valor nominal (tabla 5.1) utilizando el operador *Map*.

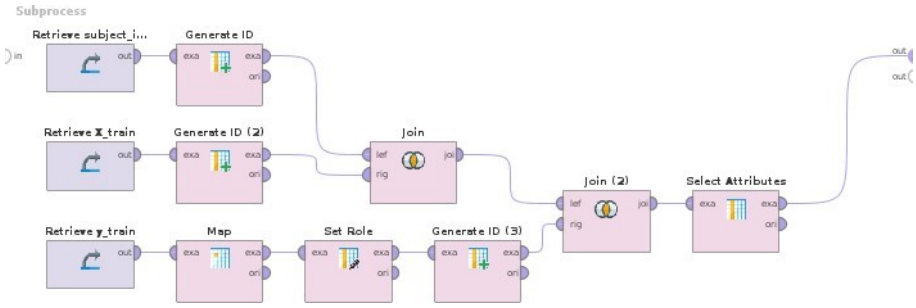

Figura 5.2: *Subprocess* de la figura 5.1 RapidMiner

El operador *Multiply* de la figura 5.1 crea copias de los subprocesos, de manera que ha cada operador (*k-NN*, *Gradiente Boosted Tree*, *W-SimpleLogistics...*), le corresponde un subproceso de datos *train* y a cada entrada del operador *Apply*

Model le corresponde un subproceso de datos *test*. El *k-NN*, el Gradient Boosted Tree y el Deep Learning tienen las mismas características que las detalladas en el apartado 4.2.2.

El clasificador *Naive Bayes*, supone que el valor de la etiqueta de cada muestra es independiente de cualquier otro atributo. A pesar de que esta suposición no suele cumplirse en la mayoría de los casos, se suelen obtener buenos resultados. Por otro lado, la simplicidad de este algoritmo puede llevar a resultados engañosos, porque si en el conjunto de entrenamiento no hay un valor de atributo para el contexto de una determinada clase, la probabilidad condicional se establece en cero, y si este valor aparece en alguno de los datos, el cero se multiplica con otras probabilidades, anulándolas y produciendo resultados engañosos. Para intentar arreglar esto, se puede activar la corrección de Laplace, de manera que se agrega uno en cada conteo y evitar que se obtengan ceros. Para la mayoría de los conjuntos de entrenamiento, agregar uno a cada recuento solo tiene un efecto insignificante en las probabilidades estimadas. Esta técnica se activa fijando a *True* el parámetro *laplace_correction*.

El clasificador *W-Simple Logistic*, construye modelos de regresión logística lineal con el algoritmo de *boosting*, *LogitBoost*, algunos de sus parámetros que se puede ajustar son:

- *I*, indica el número iteraciones para LogitBoost.
- *S*, si este parámetro se activa (*True*) se utiliza el criterio de detención en el conjunto de entrenamiento, en lugar de validación cruzada.
- *P*, si se activa este parámetro (*True*) se utiliza el error en probabilidades en lugar del error de clasificación errónea para detener el criterio.
- *M*, indica el número máximo de iteraciones de *boosting*.
- *H*, especifica el valor para detener anticipadamente LogitBoost.
- *W*, Establece beta para recorte de peso para LogitBoost. Se establece en 0 para ningún recorte de peso.
- *A*, si se activa (*True*) se utiliza AIC[2] para elegir la mejor iteración, en lugar de validación cruzada.

El clasificador *RepTree-W*, es un árbol de decisión más rápido, en el que los valores desconocidos se manejan dividiendo las instancias correspondientes en partes. Algunos de sus parámetros son:

- *M*, establece el número mínimo de instancias.
- *V*, indica la varianza mínima para la división de una clase numérica.
- *N*, fija el número de despliegues para la reducción de errores de corte.

[2]Criterio de Información de Akaike (AIC): es una medida de la calidad relativa de un modelo estadístico, para un conjunto dado de datos [45].

- *S*, especifica la semilla para la aleatoriedad de datos.
- *P*, si se establece a *True* el árbol no se poda, en cambio al establecerlo a *False* se indica que el árbol se podará.
- *L*, indica la profundidad máxima del árbol.

El valor óptimo de los parámetros principales para cada uno de los algoritmos se indica en las tablas 5.4 y 5.5.

W - Simple Logistic: 94,02 %	
I:	*0*
S:	*False*
P:	*False*
M:	*500*
H:	*50*
W:	*0*
A:	*False*

Gradient Boosted Trees: 89,41 %	
Number of trees:	*20*
Reproducible:	*False*
Maximal depth:	*5*
Min rows:	*10*
Min split improvement:	*0*
Number of bins:	*20*
Learning rate:	*0,1*
Sample rate:	*1*
Distribution:	*AUTO*
Early stopping:	*False*
Max runtime seconds:	*0*

Deep Learning: 92,16 %	
Activation:	*Rectifier*
Hidden layer sizes:	*[50; 50]*
Reproducible:	*False*
Epochs:	*10*
Compute variable importances:	*-2*
Train samples per iteration:	*False*
Adaptive Rate: *- Epsilon:* *- Rho:*	*True* *1E-8* *0,99*
Standardize:	*True*
L1:	*1E-5*
L2:	*0*
Max w2:	*10*
Loss function:	*Automatic*
Distribution function:	*AUTO*
Misssing values handling:	*MeanImputation*
Max runtime seconds:	*0*

Tabla 5.4: Valores de los parámetros óptimos de los algoritmos (1)

Naive Bayes: 74,98 %	
Laplace correction:	*True*

K-NN: 79,44 %	
k:	*20*
Measure types *- Mixed measure:*	*MixedEuclidenDistance*

W - Rep Tree: 86,21 %	
M:	*2*
V:	*0,001*
N:	*3*
S:	*1*
P:	*False*
L:	*-1*

Tabla 5.5: Valores de los parámetros óptimos de los algoritmos (2)

La tabla 5.6 engloba los resultados que se han obtenido con cada uno de los algoritmos analizados, concluyendo que el algoritmo que mejor se adapta a este conjunto de datos y por tanto, se obtiene el mejor resultado es *W - Simple Logistics*.

Familia	**Clasificador**	**Accuracy**
Lazy	*K-NN*	79,44 %
Bayesian	*Naive Bayes*	74,98 %
Trees	*Gradient Boosted Trees*	89,41 %
Neuronal Networks	*Deep Learning*	92,16 %
Linear Model	*W - Simple Logistics*	94,02 %
Trees	*W - REP Tree*	86,21 %

Tabla 5.6: Resultados de cada algoritmo con RapidMiner

En definitiva, el resultado de 94,02 % de *accuracy* que se obtiene con el modelo *W - Simple Logistics* es los suficientemente bueno para incluir la tecnología de los Smarthphone en sistemas de monitorización de la actividad.

6 | Conclusiones y mejoras

6.1. Conclusiones

Tal como se ha detallado a lo largo de este Trabajo de Fin de Grado el objetivo principal es evaluar distintos modelos que predicen la actividad diaria de una persona en una casa inteligente, a partir de datos recopilados de un conjunto de sensores. Este modelo estaría pensado como base para un sistema de monitorización que pueda servir de apoyo a personas mayores o con movilidad reducida, de modo que se les facilite llevar una vida más independiente y segura. Para ello se han analizado y evaluado dos conjuntos de datos.

El análisis y evaluación de estos dos conjuntos de datos tratados son complementarios. El primer conjunto de datos ha sido extraído, de una vivienda provista de sistema de sensores distribuidos por las diferentes estancias. Y el de la segunda contribuye como complemento, para valorar la utilización de los Smartphone, para la extracción de datos en el sistema de monitorización.

El análisis predictivo del primer conjunto se distribuye en tres fases. En la primera se construyen diferentes modelos con la herramienta RapidMiner, obteniendo los resultados que se muestran numéricamente y gráficamente en la tabla 6.1 y en la figura 6.1, respectivamente.

Familia	**Clasificador**	**Accuracy**
Lazy	*K-NN*	65,11 %
Trees	*Decision Tree*	65,04 %
Trees	*Random Forest*	70,53 %
Trees	*Gradient Boosted Trees*	74,68 %
Rule	*Rule Induction*	62,88 %
ANN	*Deep Learning*	71,48 %

Tabla 6.1: Resultados de la evaluación de los modelos con RapidMiner del primer conjunto de datos

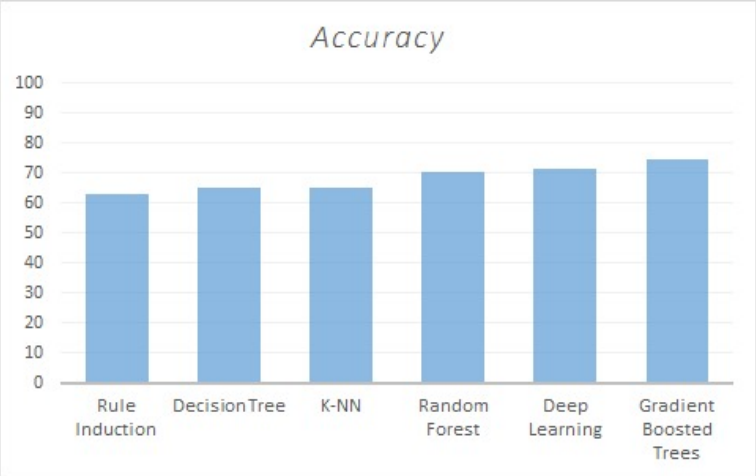

Figura 6.1: Resultados de la evaluación de los modelos con RapidMiner del primer conjunto de datos

Mediante estos experimentos se comprobó que el algoritmo que mejor se adapta a este conjunto de datos es el *Gradient Boosted Trees*. Por tanto en la segunda fase de análisis se calculan los resultados de este algoritmo usando la librería *scikit-learn* de Python, obteniendo un *accuracy* similar y un *Brier Score* (BS) de 16,6 %. La puntuación de BS obtenida se considera que es suficientemente buena, ya que, al comparar con el ranking de la competición [4], se ha alcanzado la posición 20 de 579 participantes.

En una tercera fase, se optó por reducir la dimensionalidad de los datos con el objetivo de intentar mejorar los resultados, mediante dos técnicas diferentes: la eliminación de características y la extracción de características. En la eliminación de características se eliminan las variables que se consideran redundantes, en cambio, en la extracción de características, se crean grupos de atributos independientes que son combinaciones de los atributos originales. Pero al aplicar estas técnicas en el modelo creado en Python con el algoritmo *Gradient Boosted Trees* y tal como se puede observa en la tabla 6.2, a medida que se reduce la dimensionalidad, los resultados empeoran, por tanto, en principio se descarta su utilización en el sistema de monitorización.

Eliminación de características			**Extracción de funciones**	
k	**SelectKBest *f_classif***	**SelectKBest *mutual_info_classif***	***n_components***	**PCA**
30	62,96 %	68,32 %	30	60,80 %
100	71,32 %	70,96 %	100	68,54 %
150	72,01 %	71,85 %	150	70,01 %
190	72,93 %	73,11 %	190	69,86 %

Tabla 6.2: Resultados al reducir la dimensionalidad en el primer conjunto de datos

Por otro lado, en el capítulo 5 se detallan los experimentos que se han realizado sobre el segundo conjunto de datos. Para este conjunto se ha simplificado el diseño de los modelos con la herramienta RapidMiner, y como se han obtenido buenos resultados no se ha aplicado ninguna técnica más para intentar mejorar los resultados. En la tabla 6.3 y la figura 6.2 se observan los resultados que se han obtenido con cada uno de los algoritmos, concluyendo que el mejor resultado corresponde con el algoritmo *W-SimpleLogistics*, con el que se ha alcanzado un *accuracy* de 94,02 %.

Familia	**Clasificador**	**Accuracy**
Lazy	*K-NN*	79,44 %
Bayesian	*Naive Bayes*	74,98 %
Trees	*Gradient Boosted Trees*	89,41 %
Neuronal Networks	*Deep Learning*	92,16 %
Linear Model	*W - Simple Logistics*	94,02 %
Trees	*W - REP Tree*	86,21 %

Tabla 6.3: Resultados de la evaluación de los modelos con RapidMiner del segundo conjunto de datos

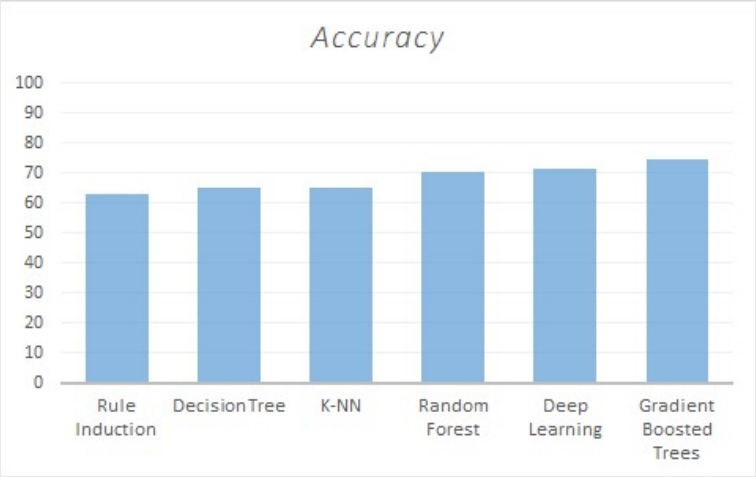

Figura 6.2: Resultados de la evaluación de los modelos con RapidMiner del segundo conjunto de datos

El resultado que se ha obtenido, con el segunda conjunto de datos, es lo suficientemente bueno para que se incluya la tecnología de los Smartphone en el diseño que se plantea más adelante en las líneas futuras de un sistema de monitorización.

6.2. Mejoras y líneas futuras

Los distintos modelos que se han evaluado en este Trabajo Fin de Grado están pensados como base para un sistema de monitorización que mediante la incorporación de nuevas soluciones tecnológicas en casas inteligentes, facilite la independencia de personas mayores o con movilidad reducida, especialmente enfocado en aquellos en riesgo de fragilidad y deterioro cognitivo leve [7]. Este sistema proporcionar una solución pro-activa de bajo coste que puede llegar a sustituir o al menos disminuir la atención humana presencial.

6.2.1. Planteamiento del sistema de monitorización

En este apartado se desarrollan las distintas partes que debe tener un sistema de monitorización basado en reconocer la posición de los sujetos, mediante un análisis predictivo con la información que recibe de los diferentes sensores eléctricos instalados previamente en las casas de los sujetos. Y en clasificar cada una de las actividades reconocidas como PELIGRO u OK. En caso de clasificar alguna de las actividades como PELIGRO es decir, extraña, arriesgada o anómala, se activaría un alerta y se contactaría con un centro de monitorización.

Las actividades que se reconocerán para detectar los posibles peligros lo más rápido posible se indican a continuación:

- De pie a tumbado
- De pie a inclinado
- De rodillas a de pie
- De tumbado a de pie
- De tumbado a sentado
- De sentado a tumbado
- De sentado a de pie
- De pie a de rodillas
- De pie a de rodillas
- De pie a sentado
- Subir escaleras
- Bajar escaleras
- En cuclillas
- Transportar algo
- Caminar
- Inclinado
- De rodillas
- Tumbado
- Sentado
- Saltar
- De pie
- Girarse

Como las necesidades, agilidades y destrezas de cada persona son diferentes, las actividades que se van a considerar peligrosas se deben personalizar. Además es importante tener en cuenta, que antes de la implantación del sistema es necesario un periodo de prueba, donde se extraen los distintos valores etiquetados para estimar que algoritmo se adapta mejor y poder completar la fase de entrenamiento necesaria para la predicción, de forma personalizado para cada usuario.

Algunas de las actividades que pueden considerarse peligrosas para una persona con capacidades reducidas es estar tumbado en cualquier lugar que no sea

la habitación durante un tiempo prolongado, lo que puede indicar una caída o un desmayo fortuito, por tanto, si se detectase esta actividad considerada peligrosa es importante tener recogidos los datos de las actividades anteriores y de esta manera, el centro de atención correspondiente al que se deriva la incidencia pueda tener una situación global de lo que y ha ocurrido rápidamente.

El diseño técnico que se plantea para el sistema monitorización requiere de la instalación de sensores ambientales, puntos de acceso y cámaras a lo largo de la casa. Además dado que se han obtenido buenos resultados al utilizar los acelerómetros del smartphone, se ha incluido su uso. Por otro lado, se instala un repositorio, que almacena y recopila la información que se extrae de los diferentes dispositivos. El sistema de monitorización se compone de cinco fases.

1. **Extracción de los datos**. Durante esta fase se recogen los datos procedentes de los acelerómetros del dispositivo triaxial colocado en la muñeca y del Smartphone colocado en la cintura del usuario monitorizado; de las cámaras y de los sensores ambientales distribuidos por las casa.
2. **Pre-procesado de los datos**. Tras la extracción de datos es necesario preprocesarlos para conseguir buenos resultados en el análisis predictivo.
3. **Análisis predictivo**. En esta fase se predicen los movimientos de los usuarios según los datos recibidos.
4. **Detección de peligro**. Durante esta fase se evalúan si los movimientos que se han predicho son peligrosos o no.
5. **Intervención comunicativa**. Si se ha detectado que el movimiento es peligroso, se inicia una vía de comunicación entre el usuario y el centro de atención correspondiente. Después si se confirma el peligro o pasado cierto tiempo en el que el centro de atención no recibe rectificación de peligro, se concluye que la existencia de peligro es real y se envía apoyo presencial al domicilio del individuo.

En la figura 6.3 se representan las distintas partes del sistema de monitorización detallado en esta sección.

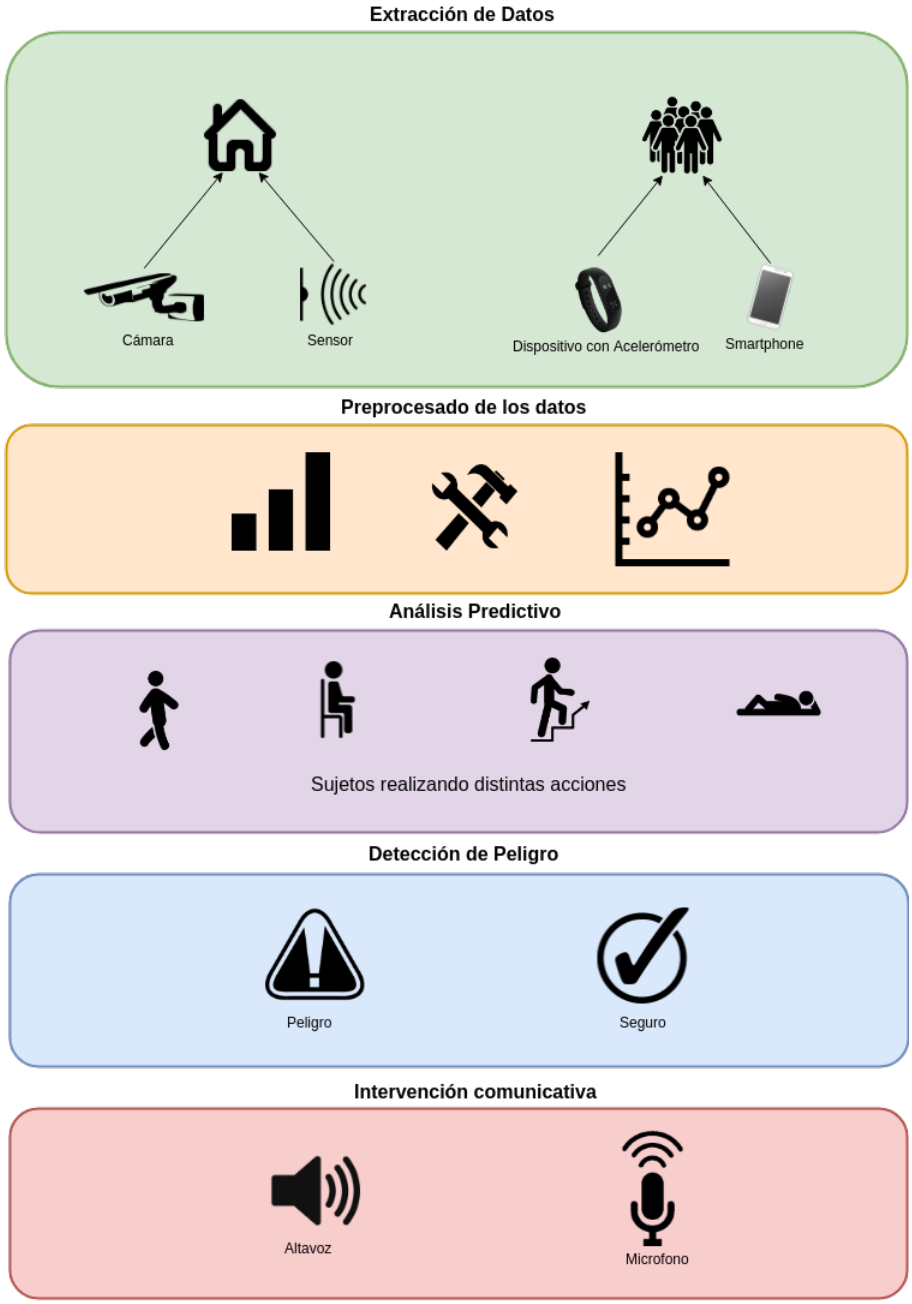

Figura 6.3: Esquema de las fases del sistema de monitorización

Entre otras, algunas de las mejores sencillas que se le podrían añadir a este sistema de monitorización son: revisar que los individuos se toman su mediación o detectar humo o gas dentro de la casa inteligente.

7 | Presupuesto

En este capítulo se desarrollará un desglose presupuestario estimado para este Trabajo Fin de Grado que recoge las tareas, recursos y costes directos e indirectos.

7.1. Planificación

El Trabajo Fin de Grado consiste principalmente en el análisis, procesamiento y creación de un modelo de inteligencia artificial que sea capaz de discernir a partir de los datos recogidos por distintos sensores, la actividad que está realizando una determinada persona en su casa. Para la ejecución del Trabajo Fin de Grado se han tenido en cuenta las siguientes partes:

1. **Documentación e investigación**. Se realizó una investigación de aquella información relativa al estado del arte del reconocimiento de actividades humanas (*Human Activity Recognition*) así como el funcionamiento y sentido de todos y cada uno de los sensores y señales involucrados y utilizados con este propósito.

2. **Obtención, análisis y pre-procesado de los datos**. Se descargaron los dos conjuntos de datos que se han utilizados. El primero fue proporcionado por una de las competiciones de la plataforma DrivenData y la segunda del repositorio *machine learning* UCI. Seguidamente estos conjuntos fueron analizados y pre-procesados

3. **Implementación de los modelos**. Se realizaron varios experimentos con cada uno de los conjuntos de datos, utilizando diferentes clasificadores y técnicas en cada uno de ellos.

4. **Pruebas**. Se evaluó cada experimento según diferentes medidas detalladas en el Trabajo Fin de Grado, para averiguar el modelo que mejor se adapta a cada conjunto de datos.

5. **Planteamiento del sistema de monitorización**. Se plantea y estructura el diseño del sistema de monitorización que se propone para detectar los posibles peligros a los que se enfrentan las personas mayores o con algún tipo de discapacidad en su día a día.

6. **Memoria**. Se redactó la memoria del trabajo realizado.

En la figura 7.1 se muestra la duración y escalonado de las tareas del Trabajo Fin de Grado.

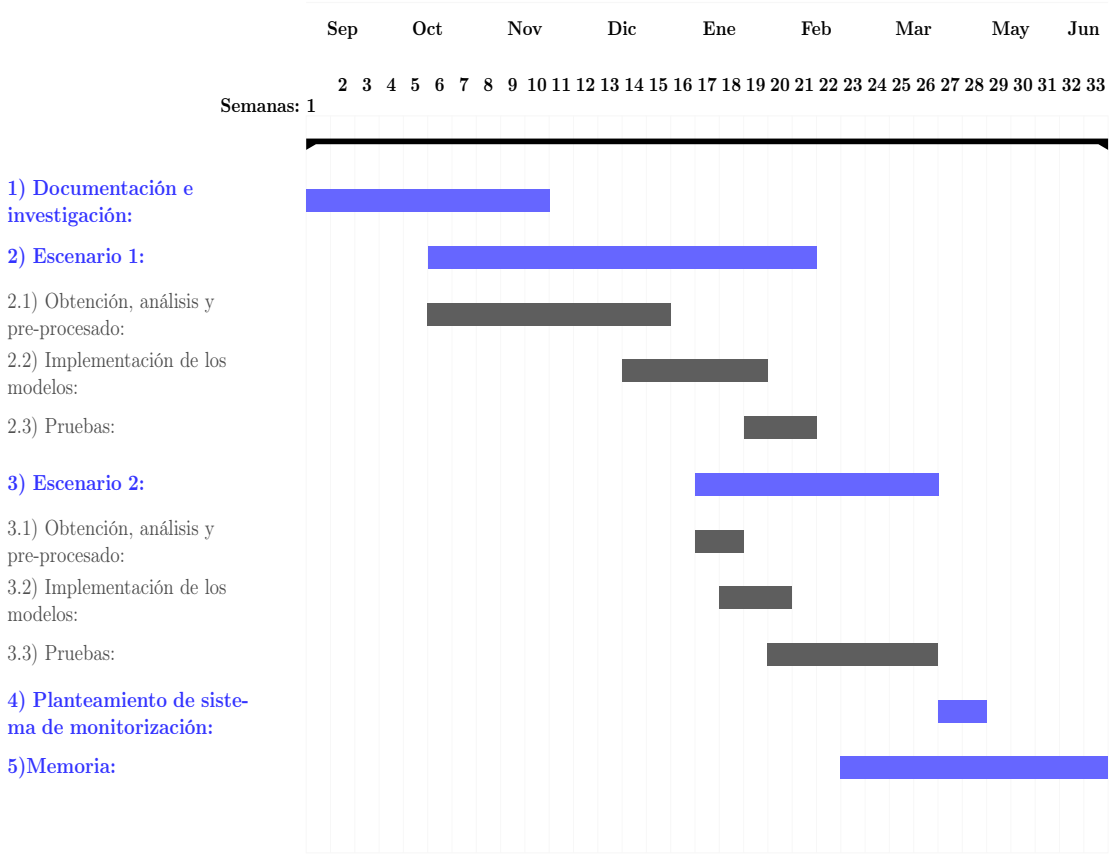

Figura 7.1: Diagrama de Gantt

7.2. Recursos y costes

En el desarrollo del Trabajo Fin de Grado han participado dos tipos de perfiles:

1. **Ingeniero Senior** (Tutor). Un ingeniero experto en técnicas de aprendizaje máquina, tratamiento de datos y reconocimiento de actividades humanas, que ha actuado como consultor funcional y se ha responsabilizado de la supervisión del Trabajo Fin de Grado y de la revisión de la documentación asociada.

2. **Analista Programador** (Alumno). Un perfil técnico con conocimientos tanto de *Python* como de sus principales librerías dedicadas al aprendizaje máquina y de *RapidMiner*. Este se ha encargado de recopilar la información sobre la metodología y el procesamiento de datos recogidos de sensores especialmente dedicados al reconocimiento de actividades humanas y de elegir la arquitectura y los frameworks a utilizar. Además ha desarrollado por completo el modelo de predicción basado en distintas técnicas de inteligencia

artificial y estimado el rendimiento futuro y las posibilidades que se pueden extraer de este modelo.

Por otro lado, se diferencian dos tipos de costes: los directos y los indirectos. Los costes directos representan el coste de recursos humanos y materiales. En cambio, los costes indirectos representan los costes derivados de la gestión y seguimiento del Trabajo Fin de Grado.

Costes directos

La dedicación en jornadas del tiempo dedicado para cada tarea especificada en el diagrama de Gant se muestra en la siguiente tabla:

Tareas:	**Dedicación Total:** (jornadas)	**Dedicación Ing. Senior:** (jornadas)	**Dedicación Analista:** (jornadas)
Documentación e investigación:	34	40 %	60 %
Escenario 1			
- Obtención, análisis y pre-procesado:	30	5 %	95 %
- Implementación de los modelos:	30	5 %	95 %
- Pruebas:	10	0 %	100 %
Escenario 2			
- Obtención, análisis y pre-procesado:	5	0 %	100 %
- Implementación de los modelos:	5	5 %	95 %
- Pruebas:	5	5 %	95 %
Planteamiento del sistema de monitorización:	5	5 %	95 %
Memoria:	40	5 %	95 %
Total: (jornadas)	164	19,35	144,65

Tabla 7.1: Dedicación en jornadas para las distintas actividades

Para el cálculo del precio por jornada y por perfil se ha tenido encuentra una ponderación basada en las tarifas de varias consultoras para los perfiles de Analista Programador e Ingeniero/Consultor Senior durante el año 2018.

El coste total de los recursos humanos dedicados al Trabajo Fin de Grado se muestra en la siguiente tabla:

	Jornadas	Tarifa jornadas	Totales
Ing. Senior (Tutor)	19,35	186,00€	3.599,10€
Analista Programador (Alumno)	144,65	70,00€	10.125,50€
		Total	13.724,60€

Tabla 7.2: Coste de los recursos humanos

Dado que el software utilizado para el desarrollo del Trabajo Fin de Grado es de libre distribución o se disponía de licencia libre de coste por motivos académicos, los únicos materiales a tener en cuenta en el cálculo de costes son dos ordenadores portátiles cuyo coste estimado se muestra en la siguiente tabla:

Descripción	Coste	Uso dedicado al TFG	Dedicación	Periodo de depreciación	Coste imputable
Portatil Desarrollo: HP Pavilion TouchSmart. Core i5. 8GB RAM. 500GB. Ubuntu 16.04.	750,00€	90 %	8 meses	60 meses	90€
Portatil Ing.Senior: HP Notebook 250 G6. Core i3. 4GB RAM. 500GB. Windowsn10 Professional.	417,00€	5 %	8 meses	60 meses	2,78€
				Total:	92,78€

Tabla 7.3: Coste de los recursos materiales

Para calcular este coste de los recursos materiales se ha empleado la ecuación 7.1 de amortización.

$$C_{Mi} = \frac{A_i}{B_i} \cdot C_i \cdot D_i \tag{7.1}$$

A = número de meses desde la fecha de facturación en que el equipo es utilizado.
B = Periodo de depreciación, 60 meses.
C = Coste del equipo (sin IVA).
D = Porcentaje de uso dedicado al Trabajo Fin de Grado.

El total de costes directos del Trabajo Fin de Grado es 13.817,38 €.

Costes indirectos

Los costes indirectos se estiman apriori y consideran los costes derivados de gestión y seguimiento del Trabajo Fin de Grado. Se calcularán considerando un porcentaje estimado en un 20 % de los costes directos. El total de costes indirectos

del Trabajo Fin de Grado es 2.763,476 €.

Cuadro resumen de costes:

Total del Presupuesto del TFG	
Total Costes Directos:	13.817,38 €
Total Costes Indirectos:	2.763,476 €
Total del Presupuesto:	**16.580,86 €**

Tabla 7.4: Presupuesto total

Bibliografía

[1] Niall Twomey, Tom Diethe, Meelis Kull, Hao Song, Massimo Camplani, Sion L. Hannuna, Xenofon Fafoutis, Ni Zhu, Pete Woznowski, Peter A. Flach, and Ian Craddock. The SPHERE challenge: Activity recognition with multimodal sensor data. *CoRR*, abs/1603.00797, 2016.

[2] vv.aa. Data science competitions to save the world. DrivenData, "`https://www.drivendata.org/`". Web; accedido el 14-06-2018.

[3] vv.aa. Welcome to the uc irvine machine learning repository. UCI, "`https://archive.ics.uci.edu/ml/index.php`". Web; accedido el 14-06-2018.

[4] vv.aa. Senior data science: Safe aging with sphere. DrivenData, "`https://www.drivendata.org/competitions/42/senior-data-science-safe-aging-with-sphere/leaderboard/`". Web; accedido el 14-06-2018.

[5] Dario Gil. Exploring the impact of cognitive computers. WIRED, "`https://www.wired.com/insights/2013/11/exploring-the-impact-of-cognitive-computers/`". Web; accedido el 14-06-2018.

[6] vv.aa. La evolución del internet de las cosas. Genexus, "`https://www.genexus.com/es/global/noticias/leer-noticia/la-evolucion-del-internet-de-las-cosas`". Web; accedido el 14-06-2018.

[7] Nicola Bryant, Nikki Spencer, Annette King, Phil Crooks, Jude Deakin, and Stuart Young. Iot and smart city services to support independence and wellbeing of older people. pages 1–6, 2017.

[8] Mukul Agarwal. Understanding the 3 vs of big data - volume, velocity and variety. Whishworks, "`https://www.whishworks.com/blog/big-data/understanding-the-3-vs-of-big-data-volume-velocity-and-variety`". Web; accedido el 14-06-2018.

[9] vv.aa. Las 5 v's del big data. Quantic Solutions, "`https://www.quanticsolutions.es/blog/las-5-vs-del-big-data/`". Web; accedido el 14-06-2018.

[10] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Comput. Netw.*, 54(15):2787–2805, October 2010.

[11] Resul Das. A survey on the internet of things solutions for the elderly and disabled: Applications, prospects, and challenges. 2017.

[12] vv.aa. Internet de las cosas. Wikipedia ,"https://es.wikipedia.org/wiki/Internet_de_las_cosas". Web; accedido el 14-06-2018.

[13] vv.aa. World population prospects. Naciones Unidas, "https://es.wikipedia.org/wiki/Internet_de_las_cosas". Web; accedido el 14-06-2018.

[14] vv.aa. Envejecimiento. Naciones Unidas, "http://www.un.org/es/sections/issues-depth/ageing/index.html". Web; accedido el 14-06-2018.

[15] Dr Sandra Wachter. Towards accountable ai in europe? The Alan Turing Institute, "https://www.turing.ac.uk/media/opinion/towards-accountable-ai-europe/". Web; accedido el 14-06-2018.

[16] Gregory Piatetsky. Will gdpr make machine learning illegal? KDnuggets, "https://www.kdnuggets.com/2018/03/gdpr-machine-learning-illegal.html". Web; accedido el 14-06-2018.

[17] Thomas W. Dinsmore. How gdpr affects data science. KDnuggets, "https://www.kdnuggets.com/2017/07/gdpr-affects-data-science.html". Web; accedido el 14-06-2018.

[18] vv.aa. Por qué necesitas empezar a tomarte en serio la protección de la información y cómo puede ayudarte eset. ESET, "https://gdpr.eset.es/". Web; accedido el 14-06-2018.

[19] Jorge-L. Reyes-Ortiz, Luca Oneto, Albert Sama, Xavier Parra, and Davide Anguita. Transition-aware human activity recognition using smartphones. *Neurocomput.*, 171(C):754–767, 2016.

[20] Albert Sama Xavier Parra Jorge-L. Reyes-Ortiz, Luca Oneto and Davide Anguita. Transition-aware human activity recognition using smartphones. UCI, "https://archive.ics.uci.edu/ml/datasets/Smartphone-Based+Recognition+of+Human+Activities+and+Postural+Transitions". Web; accedido el 14-06-2018.

[21] Jiawei Han, Jian Pei, and Micheline Kamber. Concepts and techniques. *Data mining*, 2011.

[22] Rueda Maria Solarte Jose. A proposed data mining methodology and its application to industrial engineering. 06 2002.

[23] Gregory Piatetsky. Crisp-dm, still the top methodology for analytics, data mining, or data science projects. kDnuggets, "https://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-data-science-projects.html". Web; accedido el 14-06-2018.

[24] Sergio Moro, Paulo Cortez, and Raul Laureano. Using data mining for bank direct marketing: An application of the crisp-dm methodology. 10 2011.

[25] A. M. (2009) Turing. *Computing machinery and intelligence. In Parsing the Turing Test (pp. 23-65).* Springer, Dordrecht, 1st edition, 2009.

[26] Batuhan Baykara, Martti Juhola, and Kati Iltanen. Impact of evaluation methods on decision tree accuracy. 2015.

[27] Punam Mulak and Nitin Talhar. Analysis of distance measures using k nearest neighbour algorithm on kdd dataset. *International Journal of Science and Research*, 4(7):2101–2104, 2015.

[28] G. I. Webb and Z. Zheng. Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Transactions on Knowledge and Data Engineering*, 16:980–991, 08 2004.

[29] Y. Liu and X. Yao. Ensemble learning via negative correlation. *Neural Networks*, 12(10):1399 – 1404, 1999.

[30] A. J. Canty. *Resampling methods in R: the boot package. R News, 2(3), 2-7.* 1st edition, 2002.

[31] Leo Guelman. Gradient boosting trees for auto insurance loss cost modeling and prediction. *Expert Syst. Appl.*, 39(3):3659–3667, February 2012.

[32] Ian H. Witten, Eibe Frank, and Mark A. Hall. Front matter. In *Data Mining: Practical Machine Learning Tools and Techniques (Third Edition)*, The Morgan Kaufmann Series in Data Management System, pages "i – iii. Morgan Kaufmann, Boston, third edition edition, "2011.

[33] Dr. Jose Muñoz Pérez. El perceptrón simple. `http://www.lcc.uma.es/~munozp/documentos/modelos_computacionales/temas/Tema4MC-05.pdf`. Web; accedido el 14-06-2018.

[34] Bengio Y. Courville A. Goodfellow, I. and Bengio. *Deep learning (Vol. 1). Cambridge: MIT press.* 1st edition, 2016.

[35] SHUBHAM JAIN. An overview of regularization techniques in deep learning (with python code). Analytics Vidhya, "`https://www.analyticsvidhya.com/blog/2018/04/fundamentals-deep-learning-regularization-techniques/`". Web; accedido el 14-06-2018.

[36] Cristina Scheau. Regularization in deep learning. Chat-bots life, "`https://chatbotslife.com/regularization-in-deep-learning-f649a45d6e0`". Web; accedido el 14-06-2018.

[37] Richard Dybowski, V Gant, P Weller, and R Chang. Prediction of outcome in critically ill patients using artificial neural network synthesised by genetic algorithm. *The Lancet*, 347(9009):1146–1150, 1996.

[38] Stephanie. Brier score: Definition, examples. Statistics How to, "http://www.statisticshowto.com/brier-score/", note = Web; accedido el 14-06-2018.

[39] vv.aa. Weka. Weka, "https://www.cs.waikato.ac.nz/ml/weka/". Web; accedido el 14-06-2018.

[40] vv.aa. Elan. https://tla.mpi.nl/tools/tla-tools/elan/. Web; accedido el 14-06-2018.

[41] Jason Brownlee. Why one-hot encode data in machine learning? Machine Learning Mastery Blog, "https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/". Web; accedido el 14-06-2018.

[42] vv.aa. Indicador de fuerza de la señal recibida. Wikipedia, "https://es.wikipedia.org/wiki/Indicador_de_fuerza_de_la_se%C3%B1al_recibida". Web; accedido el 14-06-2018.

[43] vv.aa. sklearn.ensemble.gradientboostingclassifier. Scikit-Learn, "http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html". Web; accedido el 14-06-2018.

[44] Sudharsan Asaithambi. Why, how and when to apply feature selection. Towards Data Science,"https://towardsdatascience.com/why-how-and-when-to-apply-feature-selection-e9c69adfabf2". Web; accedido el 14-06-2018.

[45] vv.aa. Criterio de información de akaike. Wikipedia, "https://es.wikipedia.org/wiki/Criterio_de_informaci%C3%B3n_de_Akaike". Web; accedido el 14-06-2018.

[46] Jorge Luis Reyes-Ortiz, Alessandro Ghio, Xavier Parra, Davide Anguita, Joan Cabestany, and Andreu Catala. Human activity and motion disorder recognition: towards smarter interactive cognitive environments. In *ESANN*. Citeseer, 2013.

[47] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International workshop on ambient assisted living*, pages 216–223. Springer, 2012.

[48] Davide Anguita, Alessandro Ghio, Luca Oneto, Francesc Xavier Llanas Parra, and Jorge Luis Reyes Ortiz. Energy efficient smartphone-based activity recognition using fixed-point arithmetic. *Journal of universal computer science*, 19(9):1295–1314, 2013.

[49] Jorge Luis Reyes Ortiz, Luca Oneto, Alessandro Ghio, Albert Sama, Davide Anguita, and Xavier Parra. Human activity recognition on smartphones with awareness of basic activities and postural transitions. In *International Conference on Artificial Neural Networks*, pages 177–184. Springer, 2014.

[50] Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *ESANN*, 2013.

[51] Lindsay I Smith. A tutorial on principal components analysis. Technical report, 2002.

[52] Matt Brems. A one-stop shop for principal component analysis. Towards Data Science, "`https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c`". Web; accedido el 14-06-2018.

[53] Jason Brownlee. Classification and regression trees for machine learning. Machine Learning Mastery blog, "`https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/`". Web; accedido el 14-06-2018.

[54] vv.aa. Random forest vs simple tree. Quant Dare, "`https://quantdare.com/random-forest-vs-simple-tree/`". Web; accedido el 14-06-2018.

[55] vv.aa. Descriptive, predictive, and prescriptive analytics explained. Halo Business Intelligence, "`https://halobi.com/blog/descriptive-predictive-and-prescriptive-analytics-explained/`". Web; accedido el 14-06-2018.

[56] Lidia Contreras Ochando. Prediccion e interpolacion dinamica de los niveles de contaminacion atmosferica mediante datos de intensidad de trafico y direccion del viento. 2016.